Departamento de Inteligencia Artificial
Facultad de Informática

UNA CONTRIBUCION A LA PROGRAMACION COGNITIVA: ARQUITECTURAS DE SEGUNDA GENERACION PARA REPRESENTACION DEL CONOCIMIENTO.

Autor: Manuel Alonso Gonzalez

Director de la tesis:
D. JOSE CUENA BARTOLOME

1993

TRIBUNAL ENCARGADO DE JUZGAR LA TESIS DOCTORAL:

Presidente:

Vocales:

Vocal Secretario:

Calificación:

A quienes tuvieron el arrojo de lanzarse al mar en busca de la gran ballena blanca y, para su desgracia, la encontraron.

"Los productos terminados son
para las mentes en decadencia".
Bail Channing.
Personaje de "Segunda Fundación".
Asimov, 1953.

Agradecimientos.

Como el lector sabe sobradamente, el desarrollo de una tesis doctoral es un trabajo difícil y, en sus últimas etapas, incluso angustioso. Quizá por su carácter ambicioso, esta investigación no hubiera sido posible sin la ayuda y colaboración de muchas personas. Mi más sincero agradecimiento para todas ellas.

La tesis ha sido desarrollada dentro del Programa de Formación de Personal Investigador del Ministerio de Educación y Ciencia. El trabajo se ha realizado en el marco del programa SAIH, patrocinado por la Dirección General de Obras Hidráulicas. Gracias a ambas instituciones el autor ha dispuesto de los medios humanos y materiales necesarios para llevar a cabo su labor.

Mi agradecimiento a José Cuena, director de la tesis. La investigación que da lugar a la tesis ha durado cuatro años y medio, pero hemos pasado un total de siete trabajando juntos. A lo largo de ese periodo he aprendido de él casi todo lo que se. Espero conservar su amistad para siempre y más allá de nuestras posibles diferencias.

Mi agradecimiento a las siguientes personas:

- Muy especial, a María Jesús. Ella ha sido mi mano derecha en este trabajo, ejerciendo una labor crítica y de revisión de esta memoria y empujandome, de paso, a terminar su redacción. He de agradecer, además, sus comentarios y críticas en los temas relacionados con Tareas Genéricas, de los que ella es gran conocedora. Sin su ayuda, en fin, no hubiera presentado la tesis.
- A mis padres y hermanos. Este agradecimiento no es el simple reflejo de una tradición, pero ellos ya lo saben. Espero que se sientan tan orgullosos de mí como yo lo estoy de ellos.
- A Antonio Salmerón, del que se puede afirmar que es una tesis doctoral andante ya que constituye, en sí mismo, una contribución original a este mundo. He aprendido mucho de nuestras discusiones.
- A Martín Molina, el más listo entre los miembros del laboratorio. Su gusto por la polémica y su capacidad para exprimir una discusión hasta límites insospechados, han sido para mí un reto tan continuo como provechoso. Además, su desarrollo del SAR ha hecho más fácil el diseño de mi arquitectura.
- A Eduardo Izquierdo (el *junior* que ya no lo es), por sus desarrollos alrededor del SSR y por embarcarse con decisión en nuestras últimas aventuras.
- Al resto de los estupendos colaboradores que he tenido: Pilar Ponce y Javier Llopis, por sus magníficos trabajos en interfaces gráficas; Eduardo Martín, por sus trabajos en sistemas de satisfacción de restricciones; Salvador Fontán, Jose Luis Cuena y -de nuevo- Javier Llopis, por su labor en el

desarrollo del lenguaje DECON. Finalmente, mi agradecimiento para Benito Reig y Luis Garrote, jóvenes pero ilustres hidrólogos que, en su calidad de expertos, hicieron posible la construcción de un modelo complejo en el dominio de la hidrología.

Suele ser obligado agradecer, finalmente, las ayudas recibidas en tareas de mecanografía, edición, etc. En este apartado, sin embargo, no me cabe más que agradecerme a mí mismo el haber tenido la fuerza de voluntad para llevar a cabo esa tediosa e ingrata labor.

Madrid, Enero de 1993.

Manuel Alonso Gonzalez.

Resumen.

La tesis propone el concepto y diseño de una arquitectura cognitiva para representación de conocimiento profesional especializado en clases de dominios relacionados con el mundo físico. Constituye una extensión de los trabajos de B.Chandrasekaran, potenciando el concepto de arquitectura basada en tareas genéricas propuesta por dicho autor.

En base a la arquitectura propuesta, se ha desarrollado un entorno como herramienta de construcción de sistemas expertos de segunda generación, así como un lenguaje para programación cognitiva (DECON). Dicho entorno, programado en lenguaje C sobre UNIX, ha sido utilizado para el desarrollo de un sistema para predicción de avenidas en la Cuenca Hidrográfica del Jucar, en el marco del proyecto SAIH.

Primeramente, la tesis plantea el problema de la modelización del comportamiento de los sistemas físicos, reflejando las limitaciones de las formas clásicas de representación del conocimiento para abordar dicho problema, así como los principales enfoques más recientes basados en el concepto de arquitectura cognitiva y en las técnicas de simulación cualitativa. Se realiza después una síntesis de la arquitectura propuesta, a nivel del conocimiento, para detallar posteriormente su desarrollo a nivel simbólico y de implementación, así como el método general para la construcción de modelos sobre la arquitectura. Se muestra también un resumen de los principales aspectos del desarrollo de software.

Finalmente, en forma de anejos, se presenta un caso de estudio, el sistema SIRAH (Sistema Inteligente de Razonamiento Hidrológico), junto con la gramática formal del lenguaje de soporte para la definición de modelos.

Abstract.

The thesis proposes the concept and design of a cognitive architecture for professional knowledge representation, specialized in domain classes related to the physical world. It is an extension of the Chandrasekaran's work, improving the concept of Generic Task based architecture introduced by this author.

Based on the proposed architecture, an environment has been developed, as a case of second generation building expert systems tool, as well as a language for cognitive programming (DECON). The environment, programmed in C lenguage on UNIX operating system, has been used to develop a system for flood prediction in the Jucar watershed, inside of the SAIH project.

Firstly, the behavior modeling problem of physical systems is discussed, showing the limitations of the classical representations to tackle it, beside the most recent approaches based on cognitive architecture concepts and qualitative simulation technique. An overview of the architecture at the knowledge level is then made, being followed by its symbolic and implementation level description, as well as a general guideline for building models on top of the architecture. The main aspects of software development are also introduced.

Finaly, as annexes, a case of study -the SIRAH system (Sistema Inteligente de RAzonamiento Hidrológico)- is introduced, along with the formal grammar of the support language for model definition.

Indice.

Indice de figuras.

Capítulo 7

Anejo A

1. INTRODUCCION.

La presente Tesis Doctoral tiene por objeto dar cuenta de los trabajos de investigación y desarrollo realizados por el doctorando durante el periodo comprendido entre los años 1987 a 1992, en el Departamento de Inteligencia Artificial (Facultad de Informática) y el Laboratorio de Sistemas Inteligentes (ETSICCP) de la Universidad Politécnica de Madrid. La investigación se ha encuadrado en el marco del proyecto SAIH (Sistemas Automáticos de Información Hidrológica) y ha sido patrocinada por el Ministerio de Obras Públicas y Urbanismo, mediante convenio con la Fundación Agustín de Bethencourt.

Las ideas que han contribuido a la elaboración de la tesis han sido objeto de sucesiva publicación en diferentes foros nacionales e internacionales, desde el año 1987 hasta el año 1992 (ver bibliografía). Su desarrollo se ha canalizado, además, mediante diferentes proyectos fin de carrera, entre los que cabe destacar [Fontán 91], [Izquierdo 92], [Molina 90] y [Pozo 92].

La tesis versa sobre Arquitecturas Cognitivas, debiendose entender como sinónimo de Arquitecturas de Representación del Conocimiento. La definición del término **Arquitectura** conduce a confusión ya que, si bien de sus diferentes acepciones(1) cabe inferir como definición general: **el arte de construir**, sin embargo -y ya en el dominio de la informática, se viene interpretando también en referencia al conjunto de estructuras, herramientas y métodos de soporte para hacerlo. Así, cuando se habla de Arquitectura de Computador, se habla también de características funcionales y de componentes físicos y lógicos(2).

El término Arquitecturas Cognitivas, que preside esta tesis, ha sido tomado de [Laird et al 87] donde Laird, Newel y Rosenbloom plantean la arquitectura del entorno SOAR. Curiosamente, dichos autores no lo definen, bien porque se asume como sobreentendido o, quizá, para no participar de la confusión, por otra parte ya tradicional, existente alrededor de los conceptos acuñados por la Inteligencia Artificial a lo largo de su historia [Brachman, Smith 80]. [Cuena 91] define el concepto Arquitectura Cognitiva como:

> "Conjunto de paradigmas de representación, técnicas de inferencia y métodos de adquisición del conocimiento asociados a una clase de problemas."

---

(1) Acepciones del vocablo Arquitectura en el Diccionario de la Real Academia de la Lengua:
(General): Arte de proyectar y construir edificios.
(Civil): Arte de construir edificios y monumentos públicos y particulares no religiosos.
(Hidráulica): Arte de conducir y aprovechar las aguas o de construir obras debajo de ellas.
(Militar): Arte de fortificar.
(Naval): Arte de construir embarcaciones.
(Religiosa): Arte de construir templos, monasterios, sepulcros y otras obras de carácter religioso.

(2) Diccionario Enciclopédico Tecnológico editado por Espasa Calpe (1992).

Esta definición puede aceptarse como marco de referencia en adelante. No obstante debe matizarse, en aras de una mayor precisión al centrar el tema de la tesis. En efecto, una arquitectura cognitiva debe servir para crear modelos del conocimiento, útiles para plantear y resolver problemas tal y como los plantean y resuelven las personas. Este enfoque, en linea con los objetivos de la Inteligencia Artificial, contribuye a superar la tradicional visión antropomorfica (tomando como modelo al propio computador) que ha presidido la evolución del software desde sus orígenes. Esta evolución, en sus diferentes estadios, ha consistido en sucesivas sofisticaciones de los dos conceptos característicos del software: el dato, como elemento pasivo de información, y el procedimiento, como elemento dinámico de transformación del dato. La culminación actual de ese proceso evolutivo se halla, probablemente, en los métodos y técnicas orientadas al objeto, como visión integradora de ambos aspectos.

El concepto de Arquitectura Cognitiva, tal como se plantea y desarrolla en la tesis, contribuye a ese cambio de perspectiva, al promover la definición de modelos -para resolver problemas- basados en las formas de entender y razonar de las personas, y no en las formas de computar directamente sus soluciones.

En linea con lo anterior, cabe definir el concepto **Arquitectura Cognitiva** como **el conjunto de estructuras de representación, procedimientos de interpretación y métodos de adquisición, definidos como herramientas de soporte para la construcción de modelos del conocimiento.** Por consiguiente, una arquitectura así concebida debe establecer:

1) Los tipos de conocimiento que se pueden definir y sus estructuras de soporte.
2) Los procedimientos necesarios para interpretar esos tipos de conocimiento.
3) Las reglas por las cuales se pueden organizar y manipular los dos aspectos anteriores.

Como ya se apuntó al principio, la tesis se ha desarrollado en el seno del proyecto SAIH. El objetivo del proyecto, en lo que concierne al Laboratorio de Sistemas Inteligentes, era construir un sistema para predicción del comportamiento, a corto y medio plazo, de una Cuenca Hidrográfica, en particular en presencia del fenómeno meteorológico denominado *gota fría*. Se trataba de crear una plataforma sobre la que un hidrólogo pudiera programar, a un nivel muy cognitivo, el modelo conceptual con que interpretar el fenómeno.

Este objetivo de desarrollo ha producido, como inevitable, un efecto de especialización de la investigación alrededor de los dominios relacionados con el mundo físico y su comportamiento. En consecuencia, si bien la propuesta de tesis, sus conclusiones y sus contribuciones podrían posiblemente ser objeto de generalización, deben entenderse prudentemente ceñidas al entorno de los sistemas físicos.

1.1. PLANTEAMIENTO DEL PROBLEMA.

En el mundo real existen una gran variedad de dominios conocidos por profesionales de la ingeniería que, de forma genérica, pueden encuadrarse en lo que se denominan sistemas físicos. Este tipo de sistemas, ya sean naturales o artificiales, poseen las siguientes características generales:

1) Constan de una serie de componentes interrelacionados que, conjuntamente, producen el comportamiento global del sistema.
2) Ante un conjunto de acciones externas, provocadas por el medio en el que opera el sistema o por decisiones de control adoptadas por las personas que lo gobiernan, evolucionan modificando su comportamiento, a partir de los cambios en sus elementos componentes y de las influencias de unos componentes sobre otros.
3) Entre los posibles estados alcanzables, existen estados problemáticos que requieren acciones de control por parte de los profesionales responsables.

Asimismo, la actuación de esos profesionales exige conocimiento de varios tipos [Cuena 89b]:

1) Conocimiento para analizar una situación (es decir, un estado del sistema) y establecer una valoración adecuada de la misma, en particular identificando aspectos problemáticos.
2) Conocimiento predictivo para razonar sobre la forma en que el sistema controlado puede evolucionar, de forma que sea posible identificar, en determinados horizontes temporales futuros, patrones descriptivos de sus posibles comportamientos relevantes.
3) Conocimiento de soporte para la toma de decisiones, para razonar sobre las acciones de control o de defensa más convenientes, con objeto de mejorar la operación del sistema y corregir -o, al menos, mitigar- las situaciones problemáticas detectadas en 1) y 2).

Desde el punto de vista de la representación del conocimiento, cabe afirmar que no todos los tipos de conocimiento mencionados pueden extraerse directamente de los expertos, al menos en el caso de sistemas con un alto grado de complejidad. Para el análisis de una situación y para la toma de decisiones puede pensarse, en el caso general, en esquemas clasificativos y de planificación respectivamente. Sin embargo, el conocimiento predictivo descrito en 2) se vuelve más complejo, al tratar de la forma de entender y razonar sobre el comportamiento de sistemas. En este caso es difícil que, a partir del juicio personal de los expertos, se puedan obtener directamente sus formas de razonar sobre los aspectos finales de la evolución del sistema.

Los expertos pueden aportar, no obstante:

1) Modelos que expliquen el comportamiento del sistema, así como criterios para estimar los parámetros de esos modelos.
2) Criterios para evaluar la fiabilidad de los resultados obtenidos al aplicar los modelos.

El conocimiento aportado se estructura, por consiguiente, en dos niveles: nivel básico (profundo) sobre teorías del comportamiento de un sistema, y un nivel más heurístico para interpretación y control de las mismas.

En principio, podría pensarse en un esquema de representación mixto, que recogiera con precisión el nivel básico mediante el uso de modelos clásicos de interpretación numérica, complementados con bases de conocimiento para los pasos previos de estimación de parámetros y definición de datos y los posteriores de interpretación de resultados. Esta opción ya ha sido planteada en [Alonso et al 89] [Cuena et al 91], como trabajo paralelo a esta tesis y también en el Departamento de IA de la Facultad de Informática de Madrid.

Sin embargo, este enfoque resulta excesivo y, a menudo, innecesariamente complejo ya que:

1) La realización de cálculos numéricos con mucha precisión no mejora, por sí misma, la calidad de los resultados si, como es el caso general, tanto los datos de entrada como los parámetros de los modelos aportan ya una importante cuota de error.
2) En muchos casos no se requiere un seguimiento exhaustivo y preciso de las variables del sistema. Puede ocurrir que no se necesite o, si el programa que se construye debe operar en tiempo real, no haya tiempo para ello. En estos casos, por el contrario, se requerirá un conocimiento suficiente en términos cualitativos sobre la evolución del sistema, que permita hacer conjeturas sobre los aspectos significativos del comportamiento.

Cabe plantearse, por tanto, satisfacer el nivel básico de representación para el conocimiento sobre modelos, mediante el uso de técnicas de razonamiento cualitativo, de acuerdo con las propuestas de [DeKleer, Brown 84a, 84b], [Forbus 84] y [Kuipers 86], entre otras.

El tipo de arquitectura que se propone trata de recoger ambos niveles de representación, de manera que sea posible representar conocimiento tanto de los modelos como de las formas en que los profesionales los manejan. En el primer caso mediante reducción a esquemas inferenciales del conocimiento de los modelos por técnicas de simulación cualitativa. En el segundo caso mediante esquemas de inferencia clásicos obtenidos directamente de los expertos, y todo ello programable en un lenguaje capaz de manejar conceptos asequibles a los profesionales.

1.2. OBJETIVOS DE LA INVESTIGACION.

El tipo de problemas planteado, cuyo tratamiento exige conocimiento de análisis (para diagnóstico), conocimiento para la toma de decisiones y conocimiento predictivo sobre la evolución de sistemas, sugiere un enfoque de solución en el que se conjugan las dos lineas de investigación expuestas: linea de arquitecturas cognitivas y linea de razonamiento sobre el comportamiento por simulación cualitativa.

En consecuencia, el objetivo básico de la tesis es la concepción y diseño de una clase de arquitecturas, basadas en tareas, para representación del conocimiento capaz de entender de dominios del mundo físico. La arquitectura debe englobar no solo los paradigmas de razonamiento, para resolución de problemas, del tipo uniforme (marcos, reglas, etc) sino también las formas de razonamiento, mas especializadas, utilizadas por las personas que se mueven en una determinada rama profesional.

En linea con el objetivo principal, la arquitectura debe promover la construcción de entornos, alrededor de clases de dominios, que permitan a los propios profesionales la formulación de modelos más acorde con su forma de entender la tarea de resolver problemas. Se plantea, por tanto, una forma de programación con mayor contenido cognitivo y, en consecuencia, más alejada de las formas clásicas de computar.

Los objetivos generales citados se descomponen en los siguientes objetivos parciales:

1) La arquitectura debe ser capaz de albergar los paradigmas de razonamiento sobre el mundo físico existentes, sin perderse por ello el carácter modular de la arquitectura. Es decir, se deberán definir tipos de tareas para el razonamiento cualitativo, satisfaciendo la necesidad, ya puesta de relieve por [Chandrasekaran, Milne 85], [Chandrasekaran 87] de encapsular las técnicas de simulación cualitativa e integrarlas en una arquitectura basada en tareas genéricas.
2) La representación del conocimiento de control, a todos los niveles, deberá ser explícita. Este objetivo es obligado, ya que no se puede hablar de una verdadera representación del conocimiento (ni de inteligencia) si dicha representación no incluye, explícitamente, las estrategias de uso del conocimiento. El objetivo es, en este aspecto, construir tareas específicas para representación del conocimiento de control.
3) De los objetivos anteriores se deduce la imposibilidad de utilizar ninguno de los lenguajes o entornos de representación existentes. En consecuencia, es condición necesaria la creación de un lenguaje formal propio para definición del conocimiento (en adelante lenguaje DECON) acorde con la arquitectura. Obviamente, dado que el lenguaje de representación es de nueva creación, su gramática responderá a una sintaxis pseudo-castellana. El lenguaje deberá ser eminentemente declarativo y tener carácter modular; es decir, deberá construirse por composición de las gramáticas, de cada tipo de elemento a

representar, creadas de forma semi-independiente. Ello debe permitir obtener lenguajes específicos para cada entorno diferente que se asiente en la arquitectura, sin utilizar más capacidad de representación que la requerida y evitando, en consecuencia, los efectos de sobrecarga de recursos computacionales que sufren los entornos actuales de representación del conocimiento.

Para verificar los objetivos así como la validez del tipo de arquitecturas y entornos propuestos, y teniendo en cuenta el marco de desarrollo de la tesis, se plantea como último objetivo la construcción de un entorno concreto especializado en el dominio hidrológico y la definición de un modelo operativo complejo para predicción de avenidas en cuencas hidrográficas.

1.3. ORGANIZACION DE LA TESIS.

Acorde con los objetivos propuestos, la redacción de la tesis se organiza en tres fases:

1) Antecedentes: principales ideas y trabajos anteriores, de otros autores, en los que se ha basado el proyecto investigador.
2) Desarrollo: propuesta de tesis y descripción en detalle, a todos los niveles, de la arquitectura diseñada.
3) Instrumentación: principales aspectos del diseño del software realizado.

Los antecedentes de esta tesis se hallan en dos lineas principales de investigación. De un lado, una linea de razonamiento cualitativo y, por otra parte, una linea de arquitecturas del conocimiento. Entre ambas, cabe considerar más importante la segunda, sobre la que se basa directamente el trabajo realizado. No obstante, es necesario resumir las principales técnicas existentes en razonamiento y simulación cualitativa, ya que la arquitectura presentada debe ser capaz de albergar la utilización de dichas técnicas.

La linea de arquitecturas (capítulo 2) parte de las representaciones del conocimiento de tipo uniforme, repasando los paradigmas clásicos de representación (apartado 2.1). El objetivo no es describirlos en profundidad, sino únicamente presentarlos como antecedente de las limitaciones comunes a todos ellos, que se muestran en el apartado 2.1.1. Diversos autores están de acuerdo en esas limitaciones, entre los que cabe destacar a [Chandrasekaran 83], [Winston 84, 87], [Steels 87, 89], [Cuena 89b, 91], [Garrote 90] y [Salmerón 92].

Como contrapunto a las representaciones uniformes nace el concepto de arquitectura modular, cuya principal característica es su capacidad para estructurar y dividir la tarea de la representación de los diferentes tipos de conocimiento involucrados en la resolución de problemas. En el apartado 2.2 se estudian los principales enfoques en la literatura de arquitecturas. Estos enfoques guardan entre ellos diferencias en su concepción y objetivos y, de acuerdo con ello, cabe clasificarlos de la siguiente forma:

1) Arquitecturas de implementación (apartado 2.2.1), cuyo mayor exponente son las arquitecturas de pizarra [Reddy et al. 73], [Erman et al. 80], [Baltzer et al. 80], [Erman et al. 81], [Corkill et al. 82], [Hayes-Roth 85], [Nii 86a, 86b], [Corkill et al. 88].
2) Arquitecturas cognitivas de tipo universal (apartado 2.2.2), donde se destacan las sucesivas propuestas de [Laird, Newell 83], [Laird 84], [Rosenbloom, Newell 86], cuyas investigaciones se resumen finalmente en la arquitectura del entorno SOAR [Laird et al. 87].
3) Arquitecturas cognitivas basadas en tareas genéricas (apartado 2.2.3) estudiando, en particular, los trabajos de Chandrasekaran [Chandrasekaran, Mittal 82], [Brown, Chandrasekaran 83, 84, 89], [Chandrasekaran 83, 86, 87], [Bylander, Chandrasekaran 87], [Chandrasekaran et al. 91], [Cuena, Salmerón 91] y [Chandrasekaran et al. 92].

Los trabajos de investigación enumerados son los que han servido como germen de ideas para la concepción de la clase de arquitecturas que la tesis propone. En particular, de las arquitecturas de pizarra se ha aprendido la utilización de estrategias de tipo oportunista que, como se verá más adelante, pueden ser utilizadas en el seno de las tareas, de la arquitectura, para razonamiento cualitativo. La arquitectura universal de Laird, Newell y Rosenbloom aporta como idea más aprovechable el manejo de preferencias como mecanismo de control. Finalmente, las arquitecturas basadas en tareas genéricas constituyen la base más importante de la tesis. De ellas se ha tomado el concepto de tarea como vehículo para encapsular conocimiento y estrategia para resolver problemas, si bien aplicadas al razonamiento cualitativo sobre el comportamiento de objetos físicos.

La linea de razonamiento cualitativo parte de las propuestas de [Hayes 79, 85], donde Hayes aprecia, en primer lugar, la necesidad de que la Inteligencia Artificial no se ciña en exclusiva a los "toy problems" como vehículo para poner a punto las teorías existentes y, en segundo lugar, propone una linea de investigación cuyo objetivo es establecer una teoría solida que permita comprender el mundo físico desde un punto de vista cualitativo. La idea era llegar a construir una representación del conocimiento, como soporte a esa teoría, basada totalmente en la lógica. El reto de Hayes ha sido origen de diversas investigaciones en métodos y técnicas de razonamiento cualitativo, entre las que cabe destacar las propuestas de [Kuipers 84, 85, 86], [DeKleer, Brown 84a, 84b] y [Forbus 81, 84, 88, 89]. Estas propuestas constituyen teorías rigurosas, en particular el enfoque de Kuipers, para entender el comportamiento de los sistemas físicos, si bien se alejan de la idea inicial de utilizar la lógica como lenguaje de representación. En el capítulo 3 se recoge un resumen de cada uno de estos enfoques, discutiendo las ventajas e inconvenientes de cada uno de ellos.

La fase de desarrollo se aborda de acuerdo con las ideas de [Newell 82]. Es decir, primeramente se describe la arquitectura a nivel del conocimiento (apartado 5.1). Dicha descripción se realiza, por tanto, concibiendo un agente inteligente que opera obedeciendo al principio de racionalidad. Se especifica su funcionalidad y el

conocimiento que debe poseer, pero sin entrar en detalles de la representación simbólica de ese conocimiento ni de su implementación.

La descripción a nivel simbólico se realiza en el apartado 5.2, estructurando la arquitectura en una colección de tareas. La definición de las tareas pone de relieve las necesidades de representación y, en consecuencia, sienta las bases para la definición del lenguaje de programación cognitiva DECON.

El capítulo 6 detalla la implementación de la arquitectura, presentando las soluciones de representación e inferencia para cada tipo de tarea, en el orden siguiente:

1) Tareas para resolver el problema del control (apartado 6.1). Se muestra la solución a nivel general (Tarea General de Control) y a nivel de detalle (Tareas de Generación de Escenarios).
2) Tareas para encapsular el razonamiento cualitativo sobre objetos físicos (apartado 6.2), detallando el concepto de la representación para las Tareas Básicas de Simulación.

La figura 1a muestra de forma esquemática el hilo conductor de la investigación, a partir de los antecedentes mencionados.

En el apartado 6.3 se exponen dos paradigmas clásicos de representación e inferencia, sobre los que se asienta la arquitectura, basados en marcos y reglas y en satisfacción de restricciones respectivamente. Se realiza un resumen de dos sistemas, ya desarrollados en el Laboratorio de Sistemas Inteligentes: el SAR (Sistema de Administración de marcos y Reglas, [Molina 90]) y el SSR (Sistema de Satisfacción de Restricciones, [Izquierdo 92]).

El apartado 6.4 recoge, de forma resumida por razones de espacio, las claves del desarrollo de software realizado (fase de instrumentación). Primeramente, se presenta y justifica el uso del enfoque basado en objetos como paradigma más adecuado para las tareas de diseño y programación. Seguidamente, se realiza una descripción de la implementación propiamente dicha, en el orden siguiente:

1) El apartado 6.4.1 muestra el proceso global de construcción desde un punto de vista cronológico.
2) El capítulo 6.4.2 muestra el detalle del diseño del software para tareas. Para cada tarea, se describe su estructura y especificaciones funcionales, presentando los algoritmos principales.
3) El capítulo 6.4.3 describe el diseño de los módulos reconocedores del lenguaje de definición del conocimiento (DECON), que sirve de soporte a la arquitectura. Estos módulos están basados en las herramientas yacc y lex de Unix.

La tesis propone la construcción de entornos especializados sobre el tipo de arquitecturas diseñado. El capítulo 7 describe las características que deben tener esos entornos así como la forma de construir modelos sobre ellos. Se muestra, en consecuencia, el nuevo escenario para realizar la ingeniería del conocimiento. No debe entenderse como una propuesta de metodología (que requerirá mayor investigación) si bien, paradójicamente, se razona sobre una menor necesidad de los aspectos metodológicos al abordar la tarea de adquirir conocimiento, habida cuenta del aporte cognitivo sobre el dominio que, a nivel básico, poseerán dichos entornos.

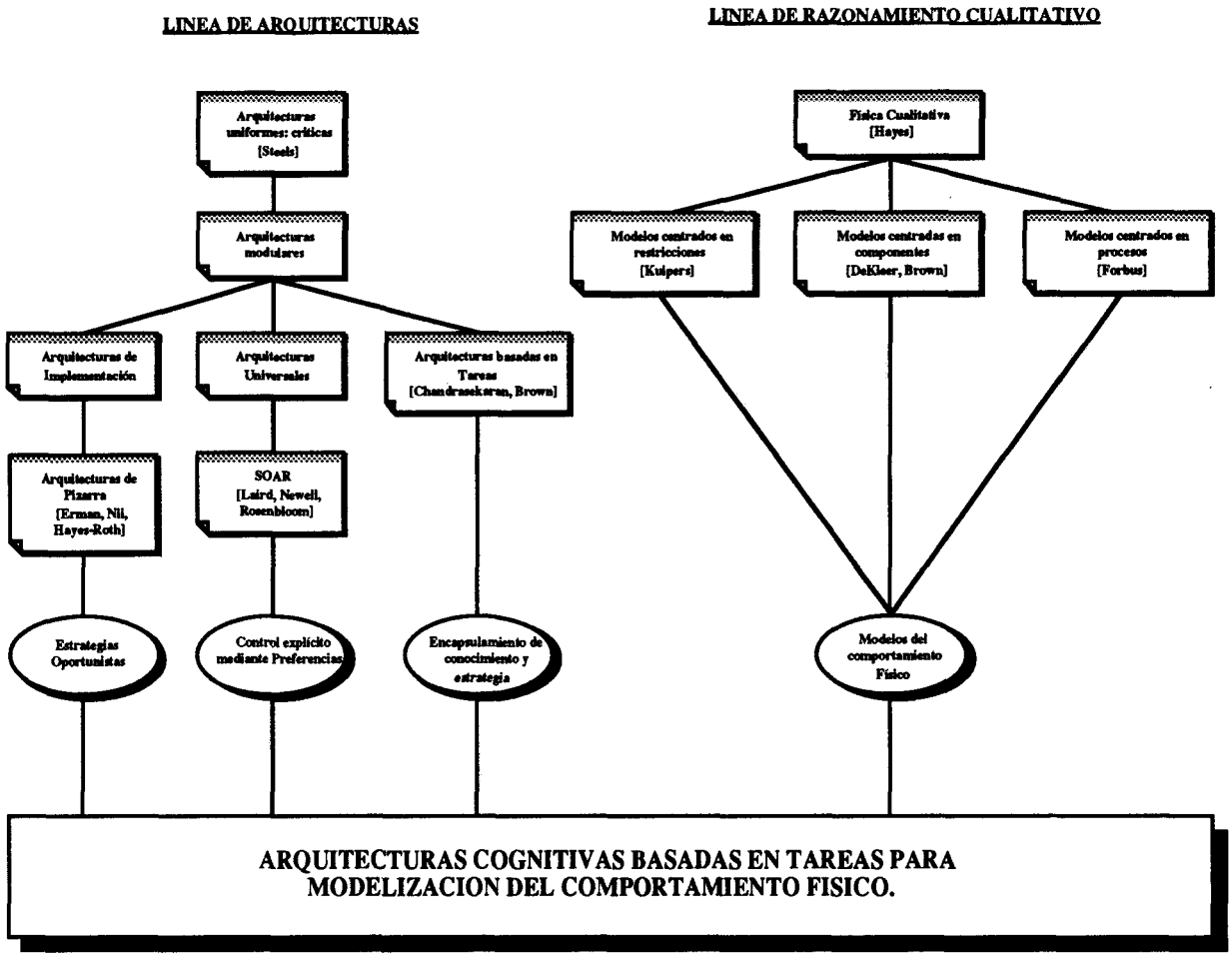


Figura 1a: Antecedentes de la investigación.

En el capítulo 8, y a modo de conclusiones, se recogen los clásicos apartados donde se repasan las contribuciones de la tesis así como, derivadas de sus limitaciones -también reconocidas, las posibles lineas futuras de continuación del trabajo investigador.

Finalmente, en forma de anejos, se resume un caso real de estudio y la gramática del lenguaje DECON. En el anejo A se describe un entorno real construido sobre la arquitectura propuesta: el entorno SIRAH (Sistema

Inteligente de RAzonamiento Hidrológico) para predicción del comportamiento, a corto y medio plazo, de Cuencas Hidrográficas, así como detalles -a título ilustrativo, de un modelo real de la cuenca del Jucar, definido sobre el entorno. Ambos, entorno y modelo, se encuentran actualmente en fase experimental instalados en la Confederación Hidrográfica del Jucar (Valencia), en linea con los sistemas de recogida de datos del SAIH.

Por su parte, el anejo B recoge la colección de gramáticas formales completas que constituyen el lenguaje DECON. El carácter modular del lenguaje permite realizar una descripción basada en gramáticas semi-independientes. De esta forma, el lenguaje se organiza realmente de manera isomorfa a la propia arquitectura. Así, se definen una gramática básica, más una gramática por cada esquema de representación del conocimiento (marcos, reglas, restricciones y cada uno de los tipos de tareas). Esas gramáticas, debidamente enlazadas y coordinadas, son generadoras del lenguaje DECON.

2. LA TRAYECTORIA DE LA REPRESENTACION DEL CONOCIMIENTO.

2.1. REPRESENTACIONES UNIFORMES.

La aparición, en la década de los setenta, de las primeras realizaciones informáticas en el área de los sistemas basados en el conocimiento, en particular de los denominados sistemas expertos, impulsó, como campo de actividad, el diseño de estructuras de información asociadas a procedimientos de interpretación cuya operación conjunta permitía resolver problemas; es decir, el diseño de representaciones del conocimiento.

Las formas de representación propuestas, desde 1956 hasta 1980, se pueden encuadrar en tres grandes grupos:

1) Reglas de producción. Este concepto nace como consecuencia de relajar la estructura de control de los algoritmos clásicos de manera que, bajo esta aproximación, un algoritmo se entiende como una sucesión de operaciones de transformación realizadas sobre una memoria de trabajo. Para un estado dado de la memoria, puede haber varias opciones de transformación, representadas mediante las reglas de producción, que se formulan mediante expresiones del tipo:

$$C_1, C_2, ..., C_n \Longrightarrow A_1, A_2, ..., A_m$$

que significan:

> Si el estado de la memoria de trabajo es tal que se satisfacen simultáneamente las condiciones $C_1, C_2, ..., C_n$ entonces deben aplicarse en secuencia las acciones $A_1, A_2, ..., A_m$.

La estrategia para obtener soluciones opera según un ciclo de control que, primero, identifica las reglas aplicables al estado actual de la memoria y, después, selecciona una (o varias) entre dichas reglas y ejecuta sus acciones, lo cual produce un nuevo estado en la memoria de trabajo. El ciclo se repite hasta alcanzar el estado de quietud, en el cual no hay ya reglas aplicables. [Forgy 82] propuso una forma de procesamiento eficiente, el algoritmo RETE, basada en este modelo, que había sido formalizado previamente por [Newel, Simon 72].

2) Reglas basadas en la lógica. En este caso la forma de razonar se plantea no como relajación de los algoritmos clásicos, sino basada en la forma de hacer demostraciones en lógica de primer orden. Su automatización parte de la regla de resolución de [Robinson 65] y culmina con la representación en clausulas de Horn y la creación del lenguaje Prolog. La definición de bases de conocimiento

constituidas por este tipo de reglas constituye una alternativa, con mayor contenido lógico, al modelo basado en reglas de producción.

3) Redes semánticas, marcos y guiones. Quillian, en paralelo con las aproximaciones anteriores, inicia su propia linea de investigación formulando un modelo en red de la memoria humana. Los nodos de esa red representan conceptos y los arcos relaciones entre conceptos. La estrategia de control para este tipo de representación consiste en la exploración de la red a la búsqueda de esquemas que encajen con el patrón definido por una pregunta dada. Esta linea originó la representación por redes semánticas, cuyo principal problema es su falta de estructuración, lo que dificulta la representación de dominios complejos de forma consistente. Estas dificultades llevan a [Minsky 75], [Minsky 81] a plantear los marcos ("Frames") como lenguaje alternativo, con mayor estructuración que las redes semánticas y más intuitivo (fácil) de utilizar que la lógica como instrumento de representación. Minsky se muestra, por tanto, partidario de crear representaciones de conocimiento verosímiles (por intuitivas) y eficientes desde el punto de vista computacional, aún a costa de no cumplir las condiciones de consistencia impuestas por la lógica.

   A pesar de la falta de rigor en su definición, los marcos suponen un mayor nivel de abstracción en la representación e introducen la noción de estructura (en el sentido de [Nilsson 80]) en el conocimiento, lo que permite pensar en formas más sistemáticas (más fáciles, por tanto) de abordar la tarea de la representación. [Hayes 77], entre otros, critican duramente los planteamientos de Minsky. Sin embargo, Hayes "legaliza" en cierto modo su utilización posterior al reducir los marcos a la lógica de primer orden. [Newel 82] sitúa esta pugna en su justa perspectiva, pues si bien admite como desaconsejable el uso de la lógica como instrumento directo de representación, reivindica su utilización como herramienta de análisis de otras representaciones que se construyan.

   Finalmente, [Schank, Abelson 77] extienden el concepto de marco y definen el guión, capaz de recoger no solo las características estáticas presentes en los marcos, sino también formas estereotipadas de operar con esas características mediante el concepto de escena.

2.1.1. Limitaciones

Las representaciones mencionadas constituyen los ejemplos más destacados del tipo uniforme (sin considerar niveles en la representación) y han sido utilizadas con éxito en una gran variedad de dominios, siendo vehículo frecuente de representación del conocimiento en la construcción de sistemas expertos. Ejemplos ya clásicos de realizaciones son los sistemas DENDRAL [Lindsay et al. 81], MACSYMA [Martin, Fateman 71], MYCIN [Shortliffe 76] y R1 [Mc Dermott 82], entre otros.

Desde esas primeras realizaciones, el concepto de sistema experto ha evolucionado, pasando de ser patrimonio de grupos investigadores a ser una realidad industrial con demanda creciente. A pesar de este innegable éxito, es necesario situar este tipo de sistemas en su justa perspectiva, con objeto de identificar sus limitaciones, que cabría considerar de dos tipos:

1) Derivadas del uso inadecuado, por trivialización, de la noción de sistema experto [Steels 87], [Cuena 91], [Garrote 90].

2) Intrínsecas a la arquitectura cognitiva utilizada, por la (a veces) difícil tarea de adquirir sobre ella el conocimiento, directamente procesable, exigido así como por la debilidad de este tipo de conocimiento para ofrecer explicaciones con la necesaria profundidad [Chandrasekaran, Mittal 82], [Winston 84], [Steels 89], [Cuena 91].

Paradójicamente, son las primeras realizaciones mencionadas las que recogen mejor el concepto de sistema experto; es decir, un programa que [Steels 87]:

1) Se diseña para asistir a un experto humano en un dominio real limitado pero difícil.

2) Contiene un modelo de la forma de entender y razonar del experto.

3) Contiene un modelo (limitado) de su propia estructura que le permite justificar su comportamiento al resolver un problema.

Ya en la década de los ochenta, el abaratamiento general de los costes del hardware, la aparición de herramientas de desarrollo sobre ordenadores de propósito general y las expectativas derivadas de los éxitos iniciales han producido la irrupción, quizá demasiado brusca, de esta tecnología en el ámbito industrial, con progresiva pérdida de rigor en su utilización. Este aspecto se acentúa en lo que respecta a la adquisición del conocimiento, actividad esencial en la construcción de este tipo de sistemas pero que, de forma creciente, se ve desplazada por aspectos de implementación, hasta el punto de considerar sistema experto a todo programa realizado en LISP, PROLOG, en un lenguaje basado en reglas o en cualquier otro lenguaje de los considerados típicos en Inteligencia Artificial.

Esta situación conduce a la utilización de la tecnología de los sistemas expertos para resolver problemas cada vez más simples que, en muchos casos incluso, podrían ser abordados adecuadamente mediante técnicas informáticas clásicas. Esto ha producido los siguientes resultado negativos:

1) Superficialidad en los resultados. Los sistemas así construidos cumplen su cometido, pero generan pérdida de credibilidad, al extenderse la creencia de que solo pueden ser aplicados a problemas triviales o muy sencillos.

2) Desencanto en la industria, al percibir que las expectativas creadas, alrededor de este tipo de sistemas, no se cumplen.

3) Relativo desencanto en un sector de la comunidad investigadora de Inteligencia Artificial, que puede considerar conveniente desmarcarse de la investigación básica en los métodos y técnicas de construcción de modelos cognitivos al constatar su trivialización en el mundo profesional.

Como consecuencia final de todo lo anterior está el hecho de que las teorías que sustentan el desarrollo actual, a nivel industrial, de los sistemas expertos se hallan en un relativo estancamiento desde principios de los ochenta. Los trabajos más recientes de [Breuker, Wielinga 85, 86], [Wielinga 87], [Chandrasekaran 87], [Chandrasekaran, Goel 88], entre otros, en identificación de métodos y técnicas de adquisición del conocimiento contradicen, afortunadamente, esta última afirmación. En efecto, estos métodos y técnicas ponen de relieve la necesidad de investigar la naturaleza del conocimiento por dos razones:

1) Traducir el conocimiento de un experto a esquemas de representación directamente procesables (por ejemplo reglas de producción) es un proceso muy difícil. El conocimiento así adquirido suele ser incompleto, cae fácilmente en la inconsistencia y resulta difícil para el ingeniero del conocimiento tenerlo bajo control, en particular según su volumen aumenta. [Chandrasekaran 83] apunta, por su parte, que estas formas "planas" de hacer inferencia no son capaces de recoger adecuadamente las peculiaridades de los procesos de resolución de problemas que realmente utilizan los profesionales de un determinado campo, estableciendo una analogía entre los paradigmas clásicos de representación y razonamiento y los lenguajes ensambladores de programación.

2) Dichos esquemas de representación tienen una capacidad explicativa limitada, pues esta se traduce en hábiles trazas de las formas deductivas utilizadas, sin hacer referencia a la naturaleza real del conocimiento empleado. [Winston 84] resalta esta realidad calificando a los sistemas basados en reglas como "sabios idiotas", en clara referencia al hecho de que las reglas de inferencia representan conocimiento condensado o superficial que descansa sobre principios básicos (teorías sobre el dominio) conocidas por el experto, pero quedando este último aspecto oculto. A veces, no basta con mostrar una regla para justificar una determinada linea de razonamiento, siendo necesario referirse a la teoría subyacente.

Finalmente, en ocasiones ni siquiera es posible adquirir conocimiento de forma que pueda interpretarse directamente por un motor de inferencia, sencillamente porque el experto correspondiente no lo tiene (y no por ello el experto deja de serlo). En dominios muy complejos (por ejemplo el mundo físico) pueden existir teorías muy sólidas que permiten comprender adecuadamente los fenómenos que tienen lugar e, incluso, disponer de un "stock" de modelos que interpretan correctamente esos fenómenos y, sin embargo, no ser posible trasladar esas teorías a esquemas heurísticos. En estos casos surge, como inevitable, la necesidad de razonar directamente a un nivel más profundo, en el cual se dote de mayor estructura a la representación del conocimiento, de manera que sea posible razonar a partir de ella.

Los paradigmas de razonamiento sobre el mundo físico de [Kuipers 86], [DeKleer, Brown 84a, 84b] y [Forbus 84, 89] son ejemplos de estas formas más profundas de representación y se comentan en el capítulo tres.

Parece claro, por tanto, que si un sistema experto debe basarse en un modelo de la forma de entender y razonar de un experto humano en un dominio, dicho modelo no debe referirse únicamente a las formas superficiales (heurísticas, derivadas de la experiencia, etc) de deducir, sino también a los principios básicos sobre los que aquellas descansan, así como al conocimiento estratégico que permite aplicar dichos principios a problemas reales.

La representación explícita de ambos niveles de conocimiento, así como la interacción entre ellos, da lugar a lo que se está denominando sistemas expertos de segunda generación [Steels 88]. Este tipo de sistemas, obviamente, imponen un cambio de arquitectura. De hecho, de la necesidad de representar conocimiento profundo (del dominio y estratégico) para dominios muy diferentes se derivará la necesidad de construir tipos de arquitecturas cognitivas también diferentes, de forma que ya no es posible hablar de una arquitectura universal para construir sistemas, sino de arquitecturas alternativas y especializadas para construir sistemas expertos en clases de problemas. Estas arquitecturas diferirán en la forma de representar teoría sobre el dominio, así como en la forma de representar el conocimiento de control.

La arquitectura que aquí se presenta incide en ambos aspectos, basandose en el concepto de Tarea Genérica propuesto por [Chandrasekaran 86, 87] como vehículo de representación de ambos tipos de conocimiento, en dominios pertenecientes al mundo físico.

2.2. ARQUITECTURAS MODULARES.

El concepto de Arquitectura Modular nace como consecuencia de las criticas vertidas sobre los esquemas clásicos de representación, en particular de su dificultad intrínseca para dotar de estructura a la representación.

Las arquitecturas modulares se caracterizan, como herramienta de soporte para la construcción de modelos cognitivos, por permitir la parcelación de los dos aspectos principales de la resolución de un problema: el conocimiento y la estrategia para usarlo. Ello debe facilitar, en general, la construcción de modelos más cercanos a como las personas entienden la resolución de una clase de problemas.

Entre las arquitecturas modulares destacan, por vías diferentes, los enfoques siguientes:

1) Las **Arquitecturas de Pizarra** [Erman et al 80], por vía de implementación, constituyendo más bien un esquema de computación modular y no tanto una arquitectura a nivel cognitivo. En este tipo de arquitecturas se plantea una forma de representar e interpretar modelos en los que coexisten diferentes **Fuentes de Conocimiento** que cooperan para resolver el problema global planteado. Las fuentes constituyen el principio de modularización, pudiendo utilizar formas de representación e inferencia propias e intercambiando información a través de una memoria central (la **Pizarra**). La decisión sobre qué fuente de conocimiento debe actuar en cada momento corresponde a un planificador, en base al estado de la pizarra y a valoraciones sobre la eficacia y oportunidad de actuación de cada fuente.
2) Las **Arquitecturas Cognitivas** del tipo **Universal**, como las presentadas en los entornos SOAR [Laird et al 87] o PRODIGY [Minton et al 87], por vía de flexibilizar y estructurar la representación. En el SOAR, por ejemplo, mediante el concepto de **Espacio Problema** y el mecanismo general de control denominado "subgoaling", todo ello conformando un esquema de resolución universal de problemas de tipo clásico GPS.
3) Las **Arquitecturas** basadas en **Tareas Genéricas** [Chandrasekaran 83, 86, 87] por la vía de dividir el método global de resolución de un problema en bloques genéricos adaptados a las clases de subproblemas en que aquel se divide.

De los tres tipos de propuestas anteriores, solo en los dos últimos casos cabe hablar realmente de arquitecturas cognitivas; es decir, de modelos de la forma en que las personas entienden la tarea de resolver problemas, mediante un esquema universal, en el caso de SOAR, y mediante esquemas especializados en el caso de las tareas genéricas. Ambos se basan en el concepto de "Knowledge Level" de [Newel 82], que defiende que la tarea de representación no debe comenzar directamente en el nivel simbólico, sino en un escalón superior estableciendo abstracciones de tipo funcional.

El primer caso (arquitecturas de pizarra), constituye más bien un modelo alternativo de computación, si bien de tipo más general, a los modelos uniformes basados en reglas y marcos.

En los apartados siguientes se presenta un resumen y discusión de cada una de las propuestas anteriores.

2.2.1. Arquitecturas de pizarra.

Los sistemas de pizarra provienen de generalizar el mecanismo de control utilizado al desarrollar el sistema HEARSAY-II [Erman et al 80]. Este sistema, orientado a la comprensión de instrucciones dadas en lenguaje hablado, fue construido en el periodo 1971-76. El concepto de pizarra, sin embargo, fue probablemente definido por vez primera de manera informal, a comienzos de los sesenta, por Newell:

> "Metafóricamente hablando, podemos pensar en un conjunto de trabajadores, todos ello mirando a la misma pizarra: cada trabajador es capaz de leer todo lo que hay en ella, y juzgar cuando él posee algo que merece la pena añadir a la misma."

La reflexión anterior se produce en relación con la rigidez de los programas clásicos (determinismo, carácter secuencial, etc) al abordar problemas de búsqueda. La solución de Newell basada en pizarras condujo a los Sistemas de Producción [Newell, Simon 72] y al sistema OPS [Forgy, McDermott 77].

Quizá la forma de comprender mejor el modelo de pizarra, en linea con las reflexiones de Newell, sea por comparación con otros modelos computacionales.

El modelo clásico de computación, por ejemplo, consiste en un programa que actúa sobre una base de datos, presentando el esquema típico entrada-proceso-salida (figura 2a). El programa contiene el conocimiento necesario para resolver el problema más el control necesario para utilizar ordenadamente dicho conocimiento.

Un modelo alternativo sería la estructura clásica de sistema experto (figura 2b). La información de entrada al sistema y los resultados se conservan en una memoria de trabajo, que es utilizada por el motor de inferencia, en conjunción con una base de conocimiento sobre el dominio, de manera sucesiva hasta que se detecta una condición de finalización. Un avance importante sobre el modelo anterior es que el conocimiento sobre el dominio se almacena como estructura procesable y separada del programa que lo manipula (motor de inferencia).

Sin embargo, el modelo tiene un importante aspecto criticable: la representación del conocimiento depende estrechamente de la naturaleza del motor de inferencia. Por ejemplo, un interprete de reglas solo puede trabajar con conocimiento expresado en forma de reglas.

El modelo de pizarra (figura 2c) viene a ser una generalización de la estructura clásica de sistema experto. En este caso, el conocimiento (tanto del dominio como de control) se segmenta en módulos, cada uno de los cuales con su propia base de conocimiento y su propio mecanismo de inferencia. Ambos, método de representación e inferencia, pueden variar de módulo a módulo. La memoria de trabajo (pizarra) se ve ahora como un medio de comunicación entre módulos. La forma de escribir y leer en la memoria debe ser, por tanto, uniforme y aceptable para todos ellos.

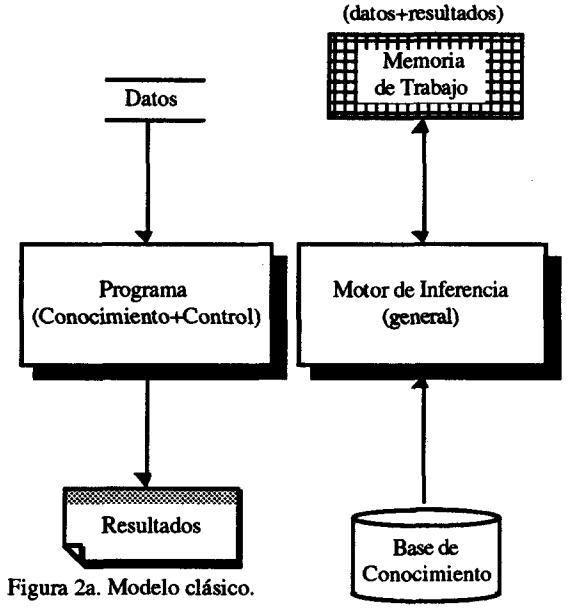


Figura 2a. Modelo clásico.

Figura 2b. Modelo clasico de SSEE.

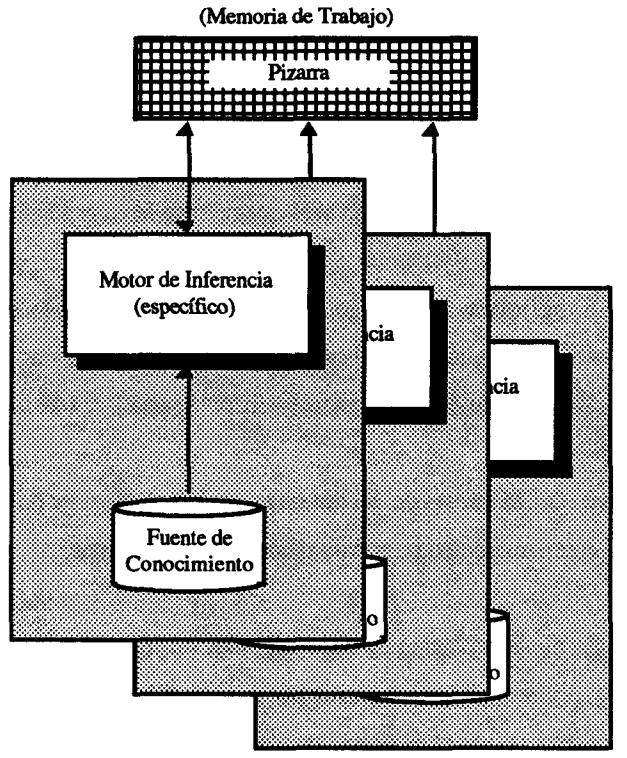


Figura 2c. Modelo de Pizarra.

El modelo de pizarra consta, pues, de dos componentes básicos:

1) Módulos especializados (denominados Fuentes de Conocimiento). El conocimiento para resolver un problema se modulariza en especialistas, en áreas concretas, separados y que operan de forma más o menos independiente.
2) Estructura central de datos (denominada Pizarra). En ella reside la definición del estado actual del resolvedor de problemas y es donde los distintos especialistas producen anotaciones encaminadas a

resolver el problema general. Las interacciones que deban producirse entre especialistas se efectuarán necesariamente a través de la pizarra.

A nivel del modelo no se especifica ningún tipo de coordinación entre los especialistas (es decir, cada especialista podría incluso entrar en juego de forma oportunista).

A continuación se describe con más detalle el sistema Hearsay-II, por ser precursor en la utilización de los modelos de pizarra.

2.2.1.1. Hearsay-II.

El proyecto Hearsay fue desarrollado, desde 1971 a 1976, en la Universidad Carnegie Mellon. Aunque tuvo una primera versión, la segunda es la comúnmente aceptada como primer sistema representativo del modelo de pizarra, y punto de partida para la cultura de los sistemas de pizarra.

El objetivo era construir un sistema que entendiera el lenguaje natural hablado. La primera dificultad estriba en el propio proceso del habla, como proceso que progresa desde el nivel de las ideas hasta la producción de sonido. En cada paso desde un nivel hacia el nivel inmediatamente inferior se pueden producir:

1) Distintas variantes todas ellas válidas.
2) Errores.
3) Ruido.

Todos estos efectos producen desviaciones respecto del mensaje ideal debidas al que habla y su entorno.

La segunda fuente de error es intrínseca al que escucha. El receptor debe realizar un proceso que se supone inverso al realizado por el que habla, recogiendo señales (al más bajo nivel) acústicas y recorriendo sucesivamente sílabas, palabras y estructuras sintácticas y semánticas de acuerdo con las intenciones del emisor. De nuevo, en cada nivel el receptor puede introducir errores de percepción y/o interpretación.

Para tolerar estos errores, un sistema que entienda lenguaje debe ser capaz de generar y evaluar numerosas interpretaciones candidatas, a cada nivel de los mencionados. Cada interpretación es denominada Hipótesis. Se tendrán hipótesis de fonemas, palabras, trozos de frases y frases completas. Lógicamente, hipótesis a un determinado nivel incorporan hipótesis de niveles más bajos.

Se preferirá una hipótesis respecto a otras que compiten con ellas si su credibilidad es alta; es decir, si su grado de consistencia con los datos en los que se apoya es mayor que el de las hipótesis que con ella compiten.

Finalmente, tres requerimientos se deben cumplir para que el sistema satisfaga sus objetivos:

1) Debe encontrar al menos un camino de hipótesis desde el nivel inferior al superior que conduzca a una interpretación correcta.
2) La interpretación dada como correcta debe tener máxima credibilidad.
3) El tiempo computacional en tiempo y espacio debe ser limitado.

Arquitectura del Hearsay-II.

Las funciones de generación, combinación y selección de hipótesis se realizan por módulos especializados: Fuentes de Conocimiento (FC). La necesidad de esta especialización viene dada por la diversidad de transformaciones necesarias para llegar desde la señal acústica hasta una interpretación válida.

Cada FC puede verse como un par condición-acción. La condición describe la situación o situaciones en las cuales la FC puede contribuir a encontrar una solución al problema planteado. La acción, por su parte, especifica en qué consiste la contribución y como se integra en el conjunto.

Ambas partes, condición y acción de una FC, son programas. Condiciones típicas serán examinar conjuntos de hipótesis y acciones típicas serán la generación de hipótesis nuevas.

Cada FC independiente se comunica con las demás a través de la pizarra, como estructura de datos global. Básicamente, la pizarra almacena las hipótesis generadas por las diferentes FCs. Cualquier FC puede generar una nueva hipótesis o modificar una ya existente.

La pizarra, por tanto, juega dos papeles: representar estados intermedios del proceso y comunicar mensajes (hipótesis) desde una FC que activa a otra FC. Para la representación de estados la pizarra se estructura en niveles, de acuerdo con los niveles identificados en el proceso de interpretación. Dichos niveles constituyen una jerarquización del espacio de búsqueda global: las hipótesis en cada nivel son una agregación o abstracción de las hipótesis consideradas a niveles inferiores, y forman un espacio de búsqueda para los especialistas operando a ese nivel, a la par que pueden restringir la búsqueda para los niveles superiores.

Dentro de este marco de trabajo, pueden considerarse dos estrategias generales de funcionamiento: descendente y ascendente, siendo el objetivo final encontrar la interpretación más creíble al más alto nivel. La estrategia

descendente reducirá el concepto general (abstracto) de sentencia en formas alternativas de sentencia, cada sentencia en secuencias alternativas de palabras etc, hasta identificar la interpretación que, en su especificación sucesiva, se ajusta mejor a los datos (acústicos) de entrada.

Por su parte, la estrategia ascendente intentará sintetizar interpretaciones a partir de los datos de entrada. Bajo esta estrategia, es posible planificar las acciones de los especialistas de manera oportunista de forma tal que, teóricamente, el sistema explote al máximo sus capacidades.

El mayor problema para descubrir la mejor interpretación (hipótesis solución al problema) estriba en la explosión combinatoria de llamadas a especialistas que puede producirse. En efecto, en cualquier momento, y a distintos niveles en la estructura de pizarra, pueden satisfacerse condiciones de múltiples FCs, dado que es imposible conseguir certeza total en el proceso de interpretación. A este fin, se utilizan métodos heurísticos con objeto de limitar las llamadas a FCs y que estas se produzcan solo en el caso de que las acciones de las FCs sean prometedoras. El mecanismo utilizado se denomina de **atención selectiva**.

El objetivo de la atención selectiva es asignar recursos computacionales (escasos) a aquellas acciones de FCs que se consideren pueden acercar en mayor medida a la solución. Esto se hace siguiendo tres criterios:

1) Estimación del efecto potencial de cada acción antes de su ejecución.
2) Estimar en qué medida es significativa una acción.
3) Estimar en qué grado es deseable ejecutar una acción en comparación con otras que con ella compiten.

El mecanismo de atención selectiva se realiza por medio de un planificador heurístico que asigna prioridades a cada acción y mantiene una cola de acciones, ejecutando en cada ciclo la de mayor prioridad y eliminandola de la cola.

La arquitectura se completa con un módulo monitor de la pizarra, que actúa como módulo de control general. Si la actividad que se ejecuta es una condición, se llevarán a la cola del planificador nuevas instancias de acciones. Por contra, si la actividad es una acción, el monitor tomará control de los cambios producidos en la pizarra y llevará a la cola de planificación las condiciones de especialistas para los que pueden ser de interés los cambios habidos.

La figura 2d muestra el esquema básico de la arquitectura del Hearsay-II.

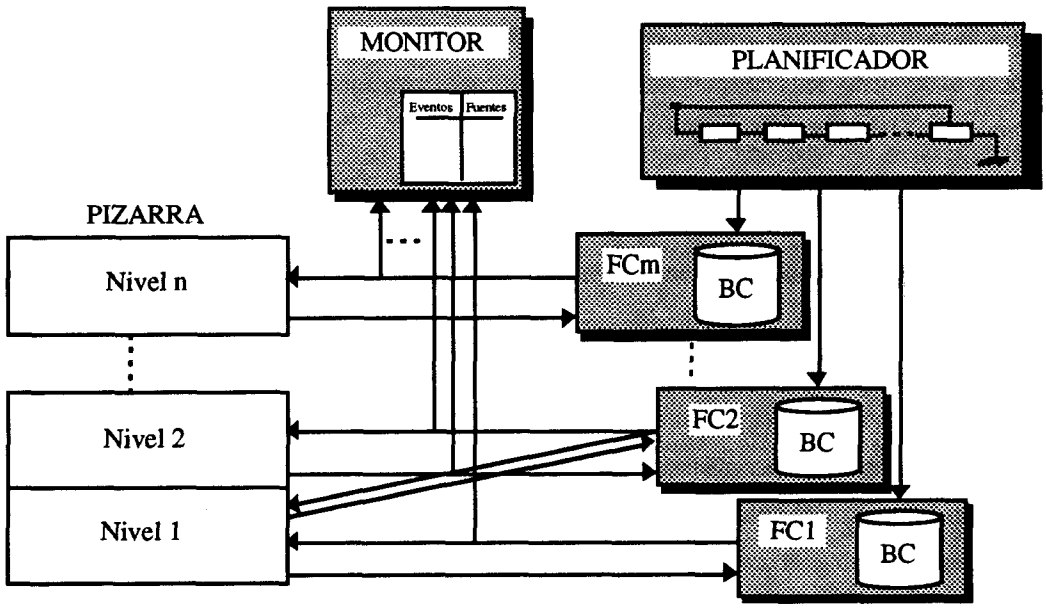


Figura 2d: Arquitectura de pizarra del Hearsay-II.

2.2.1.2. Discusión.

El apartado anterior no ha pretendido ser una descripción exhaustiva y detallada de la arquitectura de pizarra, sino reflejar su concepto básico. [Erman et al 81] establecen las siguientes conclusiones acerca de la arquitectura de Hearsay-II:

1) La resolución de problemas en terminos de generación de hipótesis y test vía fuentes independientes o semi-independientes de conocimiento más mecanismos de control centrados en el uso de una estructura global de datos, organizada en niveles de abstracción, es factible desde el punto de vista computacional.
2) La representación integrada de hipótesis en la pizarra facilita la definición del estado de la resolución del problema.
3) La representación uniforme, en los diferentes niveles de la pizarra, facilita el desarrollo y mantenimiento de las fuentes de conocimiento.
4) La estructura de control, en cuyo marco se establece la cooperación de las distintas fuentes de conocimiento, es un aspecto esencial de la arquitectura, alrededor del cual cabe hacer las reflexiones siguientes:

   - Al ser uniforme e implícita a la arquitectura, resulta fácil de entender y modificar.
   - El control de tipo oportunista total (guiado por datos y con independencia total entre fuentes de conocimiento) deriva fácilmente en problemas de sobrecarga, de manera que termina resultando

desaconsejable, salvo en casos en que la credibilidad asociada a las hipótesis generadas sea suficientemente selectiva como para estrechar fuertemente el pasillo de búsqueda.

- Facilita el uso de procesamiento paralelo en forma efectiva.

El modelo de pizarra desarrollado para Hearsay-II ha sido fuente de inspiración para investigaciones posteriores, cuyo objetivo ha sido generalizar sus principales características y producir, de una parte, entornos de desarrollo independientes del dominio y, por otro lado, con formas más sofisticadas de control. Algunas de esas propuestas se comentan a continuación.

[Corkill et al 82] hacen hincapié en que el control efectivo en una arquitectura de pizarra requiere razonar acerca de las interrelaciones entre fuentes de conocimiento que entienden de diferentes aspectos, y a diferentes niveles del problema, y que compiten por cooperar en la resolución del mismo. Esta forma de razonamiento está, de hecho, presente en Hearsay-II, pero de forma rudimentaria. Si bien ello es suficiente para los propósitos de Hearsay, el control con estrategia guiada por datos y con mecanismo directo (o instantáneo) de planificación se vuelve muy limitado al aplicar el modelo a otros dominios.

[Erman et al 81] mejoran el control por la vía de permitir a las fuentes de conocimiento interaccionar con las colas de planificación. Sin embargo, según [Corkill et al 82] este planteamiento es criticable ya que esta forma de control no se recoge explícitamente. Corkill, Lesser y Hudlicka proponen una forma de control mixto (guiado por datos y por objetivos) que permite relacionar explícitamente las actividades de las diferentes fuentes de conocimiento (FCs). Esta integración de las dos formas simples de control se basa en la idea de que la activación de un proceso de evaluación de condiciones de una fuente (bajo la forma de control guiado por datos) no solamente indica la posibilidad de ejecutar dicha fuente sino que puede ser muy deseable hacerlo, en aras de satisfacer el objetivo implícito en la salida de la FC. Para hacer estos objetivos explícitos, el conocimiento del monitor de la pizarra, que relaciona cambios en la pizarra con las FCs relevantes para esos cambios, se divide en dos partes:

1) Conocimiento de relación entre cambios en la pizarra (eventos) y posibles objetivos a satisfacer.
2) Conocimiento que relaciona objetivos con FCs.

La arquitectura se completa, bajo este enfoque, con un componente planificador que responde a la creación de objetivos desarrollando planes que los satisfagan. De esta forma es posible establecer formas de control diferentes, desde control puro guiado por datos hasta planificación de acciones guiadas por objetivos.

Esta aproximación de Lesser y Corkill, junto con otras investigaciones que abundan en el problema del control llevan progresivamente a considerar una evolución de la arquitectura básica de pizarra hacia arquitecturas derivadas

con representación explícita de control. Por ejemplo, en el diseño de Hearsay-III, [Erman, et al 81] especifican en la arquitectura una pizarra separada para control. Asimismo, [Hayes-Roth 85] define detalladamente las implicaciones teóricas de este planteamiento. En esta misma linea de generalizar el modelo de Hearsay-II se situarían los sistemas HASP y AGE.

Con posterioridad a estos trabajos se han realizado aplicaciones, tanto en dominios concretos como en la producción de entornos de desarrollo apoyados en la arquitectura, aportando flexibilidad y eficiencia [Corkill et al. 88], así como integración de diversas formas de razonamiento para control [Nii 86a, 86b], etc, pero sin aporte relevante al esquema inicial desde el punto de vista cognitivo.

En resumen, las arquitecturas de pizarra se pueden definir como arquitecturas generales y, por tanto, flexibles para implementación; es decir, un conjunto de módulos que cooperan de acuerdo con unos determinados protocolos, pero sin paralelo cognitivo. Su concepto se puede entender más próximo a una forma flexible de computar que a una forma de pensar, donde debe haber restricciones que representen la forma de entender sobre una clase de problema o sobre un dominio. Por el contrario, una arquitectura cognitiva responde a lo que [Newell 82] denomina una especificación a nivel cognitivo ("knowledge level"), que se traduce en una topografía del conocimiento donde no se detalla cómo ese conocimiento es computado sino cómo es entendido. Desde el punto de vista informático, este aspecto puede reconocerse como no novedoso, como el propio Newell admite, ya que coincide en gran medida con el concepto de abstracción funcional.

Precisamente Newell, en su famoso articulo "The Knowledge Level", apunta el hecho de que las arquitecturas de pizarra constituyen el primer tipo de arquitectura que se aproxima al principio del análisis a nivel del conocimiento. Esto puede aceptarse así si las arquitecturas de pizarra se entienden como un modelo de cooperación entre inteligencias; es decir, como modelo de cooperación de distintos elementos cognitivos con el objetivo común de resolver un problema. Sin embargo, hasta que surgió el concepto de cooperación la arquitectura de pizarra se podía considerar básicamente como una memoria de trabajo sobre la cual diferentes bases de conocimiento (en sentido general) intercambiaban información. Eran, por tanto, modelos de la forma de computar y no tanto de la forma de entender. En este sentido, pues, las arquitecturas de pizarra deben considerarse como arquitecturas de implementación.

No obstante lo anterior, este tipo de arquitecturas puede ser un vehículo muy adecuado para implementar modelos cognitivos; por ejemplo modelos basados en agentes, asignando una fuente de conocimiento para implementar el comportamiento que debe exhibir cada agente y el modelo de cooperación mediante un planificador. Esto sería una forma de representar a nivel simbólico y de implementación (arquitectura de pizarra) la definición previa a nivel cognitivo (agentes y modelo de cooperación).

Cabe concluir, pues, que las arquitecturas de pizarra sirven mejor como vehículo de implementación que como una forma de tipificar conocimiento.

2.2.2. Resolución universal de problemas: SOAR

SOAR [Laird et al. 87] es un sistema que implementa una arquitectura universal de resolución de problemas. Su modelo es el de un agente general capaz de:

1) Trabajar en todo tipo de tareas.
2) Emplear todo tipo de métodos de resolución de problemas y representaciones.
3) Aprender de las tareas que realiza.

SOAR ha sido aplicado a un amplio abanico de problemas, desde los típicos "toy problems" hasta los clásicos sistemas expertos ya instalados industrialmente, donde cabe destacar el caso de R1-SOAR, reescritura del sistema R1 [McDermott 82], y una versión de NEOMYCIN [Clancey 83] entre otros.

Cabe situar el origen de SOAR en una estructuración de las ideas de GPS [Newell, Simon 63], [Ernst, Newell 69], aunque sus autores lo sitúan de forma más general en la teoría general de resolución de problemas [Newell, Simon 72].

El principio básico de estructuración del conocimiento es el Espacio Problema. Un espacio problema es una unidad cognitiva en la que se localizan una serie de objetivos así como el conocimiento para resolverlos. La operación de un espacio problema se produce al establecer un estado inicial y un estado objetivo. El conocimiento básico para transitar de uno a otro estado se formula mediante reglas de producción. El conjunto de espacios problema constituye la representación simbólica de un agente inteligente (el modelo global SOAR) que, de esta forma, se estructura en una colección de subagentes con una estrategia de cooperación. Esta estrategia se realiza mediante el mecanismo de "subgoaling", como forma general de descomponer un problema en subproblemas.

En SOAR, la tarea de satisfacer un objetivo se formula como alcanzar un estado deseado dentro de un espacio problema aplicando los operadores (producciones) propios del espacio. La operación dentro de un espacio problema toma la forma, por tanto, de búsqueda heurística:

1) Dado un problema definido por un estado inicial y un objetivo (estado deseado) se busca un espacio problema en que resolverlo.
2) Se opera la base de conocimiento del espacio seleccionado.

3) Si el espacio, en el curso de su operación, se encuentra ante una decisión que se convierte en problemática (no se tiene conocimiento inmediato -en el propio espacio problema- para tomarla; por ejemplo, no se satisfacen las precondiciones de un operador que se quiere aplicar, o la propia aplicación resulta muy complicada) se definen nuevos subobjetivos que se intentarán satisfacer en otros espacios problema.

La definición de subobjetivos puede darse tanto para dividir un problema de partida en el dominio como para tomar decisiones de control ("universal subgoaling"). Esto permite exhibir a SOAR su comportamiento explícitamente.

El conocimiento, a nivel básico, se representa mediante reglas de producción. Esto es aplicable tanto al conocimiento de control como al de acción. SOAR emplea un sistema de producción especializado, versión simplificada de OPS5, en que la estrategia básica es más sencilla (no hay resolución de conflictos, todas las reglas que se satisfacen se disparan) y las propias producciones son más limitadas (solo pueden añadir elementos a la memoria de trabajo, no borrarlos ni modificarlos).

Una característica importante de SOAR es su capacidad de realizar aprendizaje mediante el mecanismo de "chunking" [Rosenbloom, Newell 86], a través del cual se capturan los resultados de satisfacer subobjetivos como producciones de forma automática, de manera que si el sistema vuelve a investigar problemas similares, ya no se produzcan situaciones de "impasse" que den lugar al establecimiento de subobjetivos a resolver en espacios problemas subordinados.

2.2.2.1. Arquitectura de la implementación.

SOAR se define como una arquitectura para resolución de problemas, vertebrada alrededor del espacio problema como forma de tipificar conocimiento. Se concibe, por tanto, para representar diferentes formas de entender la resolución de problemas, y no como una arquitectura para manipulación simbólica. Es desde este punto de vista desde el cual SOAR puede entenderse como una arquitectura a nivel cognitivo. De hecho, en diferentes versiones de SOAR (SOAR1 [Laird, Newell 83], SOAR2 [Laird 84b]) se han ido realizando diferentes métodos generales ("weak methods") de resolución de problemas, sin más que formalizar el conocimiento de cada tipo de método en base a la definición de los espacios problema y las funciones de control apropiadas a cada caso. Esto hace que SOAR pueda exhibir el comportamiento de cada uno de esos métodos ("universal weak method" [Laird, Newell 83]).

Esa vocación de universalidad hace que la descripción de la arquitectura, a nivel simbólico, sea muy sencilla; Así, el comportamiento dentro de un espacio problema queda completamente definido por dos conjuntos de funciones (en el sentido general):

1) Funciones para implementación de tareas; es decir, funciones para la consulta y/o generación de espacios problema, operadores dentro de cada espacio y estados de todo tipo.
2) Funciones para el control de la búsqueda; es decir, funciones para seleccionar un espacio problema, un estado o un operador para su aplicación.

El conocimiento para realizar ambos tipos de funciones define el comportamiento inteligente.

A nivel de implementación, la arquitectura se define mediante dos estructuras:

1) Una memoria de trabajo, que alberga distintos tipos de objetos que, conjuntamente, definen el estado del proceso de resolución.
2) Una estructura de proceso, que implementa las funciones de control del proceso de búsqueda.

La memoria de trabajo consta de los siguientes componentes:

1) Una pila de contextos de trabajo que especifica objetivos activos -pendientes de satisfacer-, espacios problema -pendientes de explorar-, estado y operador asociado. La cima de la pila contiene el contexto cuyo objetivo asociado es el de más alto nivel. Sucesivamente, los objetivos asociados a contextos en niveles inferiores de la pila serán subobjetivos de aquel.
2) Un conjunto de objetos para representar cada elemento de un contexto. Los objetos son, básicamente, conjuntos de ternas objeto-atributo-valor (donde el valor referencia, a su vez, a un objeto, etc).
3) Un conjunto de objetos preferencia para representar valoraciones de las posibles asignaciones a realizar a los elementos de un contexto.

La estructura de proceso recoge la funcionalidad requerida para efectuar la búsqueda en un espacio problema, utilizando conocimiento de dos tipos:

1) Conocimiento en relación con la tarea a realizar y que permite añadir nuevos objetos a la memoria de trabajo.
2) Conocimiento en relación con el control de la búsqueda, para seleccionar entre varios objetos alternativos. Una acción de control siempre toma la forma de sustitución de un elemento de un contexto por otro (en el caso más simple el elemento previo es vacío).

El ciclo general de inferencia opera en dos fases denominadas Elaboración y Decisión. La fase de Elaboración se plantea como un conjunto de producciones de la forma:

Si $C_1, C_2, ..., C_n$ **Entonces Añadir** $A_1, A_2, ..., A_m$.

donde C_i son condiciones a examinar en la memoria de trabajo, en tanto que A_i son acciones que generan nuevos objetos, valores de atributos, y preferencias. Una producción genera una instancia válida si, a partir de la memoria de trabajo, es posible establecer una asignación consistente de variables de manera que el conjunto de condiciones de la producción se satisfaga. Obviamente, pueden existir diferentes asignaciones válidas, dando lugar a otras tantas instancias de la producción y aptas para su disparo. El conjunto de todas las instancias generadas de todas las producciones se disparan simultáneamente. Esto quiere decir que en SOAR no existe fase de resolución de conflictos propiamente dicha, a excepción del lógico mecanismo de refracción, que impide que una instancia de una producción se dispare más de una vez. Finalmente, el ciclo de evaluación de producciones y el disparo de instancias generadas se repite hasta alcanzar el estado de quietud (no es posible generar más instancias de ninguna producción).

De la estructura de una producción se desprende que la fase de elaboración constituye un proceso de tipo monótono, ya que no se borran ni modifican elementos de la memoria de trabajo. La monotonía se produce solamente desde el punto de vista sintáctico, no controlándose el hecho de que la ejecución de producciones puede dar lugar a conflictos de tipo semántico. Esto es así debido a que cada producción expresa una fuente independiente de conocimiento, por tanto sin restricciones sobre lo que pueda concluir.

La fase de decisión se ejecuta cuando la fase de elaboración alcanza el estado de quietud. Su misión principal es revisar los contenidos de los contextos activos en la memoria de trabajo, decidiendo qué elementos deben sustituirse y por qué valor. Sobre cada elemento se razona en base al conocimiento sobre preferencias existente en la memoria de trabajo al final de la fase de elaboración. La interpretación de preferencias dará lugar a la sustitución del valor de un elemento de contexto. Ello equivaldrá a la adición de un nuevo tipo de objeto como valor para ese elemento (cuando previamente no existía ninguno) o al reemplazamiento por un objeto nuevo (cuando existía objeto previo). Este proceso se lleva a cabo para cada contexto abierto, desde el más antiguo al más nuevo (es decir, desde el objetivo de más alto nivel hasta el de nivel más bajo). Dentro de cada contexto, los elementos se estudian en orden: primero el espacio problema, después el estado y, finalmente, el operador. Cuando se introduce un valor en un atributo de contexto, todos los de rango inferior quedan pendientes de decisión. Esta ordenación de los atributos de contexto ya constituye una preferencia en sí misma: se asume siempre como preferible realizar cambios a alto nivel.

Una preferencia es una afirmación acerca de la selección de objeto para un atributo (o conjunto de atributos) de un contexto. Las preferencias se basan en tres primitivas:

1) Aceptable: el objeto es un candidato para actuar como valor de un atributo de contexto (espacio problema, estado u operador).
2) Rechazable: el objeto no debe ser seleccionado.
3) Deseable: el objeto es puesto en comparación con otros objetos de referencia, resultando ser mejor (más deseable para ocupar el valor del atributo de contexto), peor o indiferente.

Los objetos sobre los cuales existan preferencias que asignen para ellos valores **aceptable** y **no rechazable** formarán el conjunto de objetos candidatos a ocupar plaza en un atributo de contexto. El carácter deseable, por su parte, establece un orden de selección en ese conjunto.

La simple aplicación de los conceptos anteriores, sin embargo, puede ser en ocasiones insuficiente para tomar una decisión. Por un lado, el carácter independiente de las producciones en fase de elaboración puede dar lugar a que un objeto sea considerado aceptable y rechazable al mismo tiempo (en preferencias distintas). Para tomar una decisión al respecto, no es posible instituir:

$$\forall x \, (\text{aceptable}(x) \leftrightarrow \neg \, \text{rechazable}(x))$$

ya que ello conduciría a una contradicción lógica. El problema se resuelve dando prioridad al carácter rechazable de un objeto, eliminandolo en este caso del conjunto de objetos bajo consideración. La independencia de las producciones también puede producir conflictos respecto del carácter deseable de un objeto. Es decir, un objeto puede resultar mejor y peor que otro al mismo tiempo. Ello obliga a introducir una caracterización adicional para cada objeto estableciendo que, dados los objetos x e y, x **domina** a y si x es mejor que y no ocurriendo que y sea mejor que x.

Por otro lado, la fase de decisión también tiene que hacer frente al carácter incompleto de la fase de elaboración. Efectivamente, las preferencias entregadas por la fase de elaboración pueden no ser suficientes para decidir; por ejemplo, un conjunto de preferencias puede establecer que x es mejor que y así como que y es rechazable, pero no decir nada acerca de si x es o no aceptable. En resumen, el conocimiento puede ser (y debe poder ser) incompleto. Para que sea posible, en cualquier caso, tomar una decisión, se asume que cualquier asignación de preferencia para establecer que un objeto es aceptable, rechazable, mejor que otro, etc, debe ser realizada de forma explícita, a no ser que pueda ser deducida de otra preferencia explícita (por ejemplo, todo objeto es no aceptable, a menos que exista una preferencia que explícitamente afirme lo contrario; asimismo, si x es indiferente a y e y es indiferente a

z, entonces -implícitamente- x es indiferente a z, etc). La tabla adjunta recoge la semántica completa de definición de preferencias [Laird et al 87].

Predicados y funciones sobre objetos:

actual: el objeto que actualmente ocupa el *sl ot*
aceptable(x): x es aceptable.
rechazado(x): x es rechazado.
(x>y): x es mejor que y.
(x<y): x es peor que y (lo mismo que y>x).
(x ≈ y): x es indiferente a y.
(x>>y): x domina a y: (x>y) y ¬ (y>x).

Formulas de referencia:

indiferente(x) => ∀y [indiferente(y) => (x ≈ y)].
el-mejor(x) => ∀y [el-mejor(y) =>(x ≈ y)] y [¬ el-mejor(y) y ¬ (y>x) => (x>y)].
el-peor(x) => ∀y [el-peor(y) =>(x ≈ y)] y [¬ el-peor(y) y ¬ (y<x) => (x<y)].

Propiedades básicas:

(x>y) es transitiva, pero no completa ni anti-simétrica.
Indiferencia es una relación de equivalencia y admite sustitución sobre >:
(x>y) y (y ≈ z) impli ca (x>z).
Indiferencia no admi te sustitución en aceptable, rechazado, el-mejor y el-peor:
aceptable(x) y (x≈y) no implica aceptable(y).
rechazado(x) y (x≈y) no impli ca rechazado(y), y lo mismo para el resto.

Cierre lógico:

Las preferencias no explici tamente mencionadas y no impli cadas por transitividad o susti tuci on no se asumen como ciertas.

Definiciones intermedias:

alternativas-consideradas = { objetos ∋ x / aceptable(x) y ¬ rechazado(x) }.
maximo(X) = { X ∋ x /∀y ¬ (y >>x) }.
maximo-al ternativas = maximo(alternativas-consi deradas).
vacio(X) = ¬∃ x / X ∋ x.
mutuamente-indi ferente(X) <=> ∀ x,y / X ∋ x,y (x ≈ y).
aleatorio(X) = eli ge un elemento de X de forma aleatoria.
selecciona(X) = Si X ∋ actual, entonces actual, sino aleatori o(X).

El eccion final:

vacio(maximo-al ternativas) y ¬ rechazado(actual) => eleccion-final (actual).
mutuamente-indi ferente(maximo-alternativas) y ¬ vacio(maximo-al ternativas)
=> el eccion-final(selecciona(maximo-alternativas)).

Bloqueo (*impasse*):

vacio(maximo-al ternativas) y rechazado(actual) => *impasse*.
¬ mutuamente-indi ferentes(maximo-alternativas) => *impass e* (maximo-alternativas).

Semántica del manejo de preferencias en SOAR.

Para cada atributo, el proceso de preferencias se resume de acuerdo con los pasos siguientes:

1) Se determina el conjunto de objetos **candidatos**. Son objetos candidatos aquellos con carácter aceptable y no rechazable (es decir, existen preferencias que les asignan valor aceptable y no existen preferencias que les asignen valor rechazable).
2) Para cada objeto candidato se determina su carácter dominante, en base a las siguientes consideraciones:
   - Un objeto x domina a otro objeto y si existe una preferencia que le asigna una carácter mejor que y (actuando y como objeto de referencia) y no existe preferencia que asigne a y como mejor que x.
   - Un objeto x asignado como **el mejor** (ver semántica de preferencias) domina a cualquier otro objeto para el que no exista esta asignación, excepto aquel para el que exista una preferencia que le haga explícitamente mejor que x.
   - Un objeto x asignado como **el peor** (ver semántica de preferencias) es dominado por cualquier otro objeto para el que no exista esta asignación, excepto aquel para el que exista una preferencia que le haga explícitamente peor que él.
3) Los objetos no dominados por ningún otro objeto forman el **conjunto máximo.**
4) La fase de decisión elegirá del conjunto máximo el objeto que ha de ocupar el valor del atributo en estudio. El valor previo del atributo actúa como asignación por defecto, de manera que solo será sustituido si existe otra opción explícitamente mejor para ocupar la plaza. Así, si el conjunto máximo es vacío y el valor actual no es rechazado por ninguna preferencia, se mantendrá el valor actual. Si el conjunto máximo es no vacío, y sus objetos son **mutuamente indiferentes**, uno de ellos será la opción elegida, según el siguiente esquema:
   - Si el valor actual pertenece al conjunto máximo, él será el elegido.
   - En caso contrario, la opción final se elige aleatoriamente del conjunto máximo.

Si la fase de decisión sustituye finalmente el valor del atributo, los atributos de rango inferior quedan pendientes de decisión de cara a un próximo ciclo de inferencia. Si el valor actual del atributo se conserva, la fase de decisión pasa a repetir el proceso para el atributo de rango inmediatamente inferior.

Puede ocurrir que, finalmente, la fase de decisión no elija objeto entre los considerados, alcanzándose una situación de bloqueo ("impasse"), que puede deberse a cuatro posibles razones:

1) Empate: hay objetos que compiten por ocupar la plaza de valor del atributo pero no hay suficiente conocimiento para discriminar. Esta situación se presentará cuando existan objetos del conjunto máximo que, sin entrar en conflicto, sin embargo no sean mutuamente indiferentes.

2) Conflicto: hay objetos en el conjunto máximo que entran en conflicto.
3) Sin cambios: prevalece el valor por defecto de cada atributo.
4) Rechazo: el conjunto máximo es vacío y el valor actual se rechaza.

La forma de deshacer un bloqueo será la definición de un subobjetivo, en un nuevo contexto, proponiendo la transición a un estado que supere la situación de bloqueo.

Como último aspecto relevante de SOAR cabe mencionar su mecanismo de aprendizaje ("Chunking"). Es una adopción del mecanismo presentado en [Rosenbloom, Newell 86] y puede utilizarse siempre que se genera un objetivo nuevo del tipo que sea; es decir, como resultado de una situación de bloqueo. Consiste básicamente en crear producciones que son capaces de resumir el proceso de satisfacción de un objetivo dado. Las acciones de las producciones aprendidas generan los elementos necesarios en la memoria de trabajo para deshacer el bloqueo que dio lugar a la creación del subobjetivo. Por su parte, las condiciones exigirán aquellos aspectos relevantes para la ejecución de esas acciones.

La información necesaria para construir una producción se recaba por análisis de elementos de traza generados durante el proceso del subobjetivo y una vez terminado este. Cada elemento de traza se corresponde con una producción examinada y disparada y contiene:

1) Elementos de la memoria de trabajo que cumplen los patrones impuestos por las premisas de la producción (**elementos de condición**).
2) Elementos generados por las acciones de la producción (**elementos de acción**). Si la producción no aporta ningún nuevo elemento a la memoria de trabajo no se genera traza para ella.

Cada traza se asocia con el objetivo para satisfacer el cual se dispara la producción (el objetivo de más bajo nivel en la pila de contextos que es referenciado por condiciones de la producción). Cuando el proceso del subobjetivo finaliza, sus resultados (elementos añadidos a la memoria de trabajo) se dividen en grupos independientes; es decir, grupos cuyos elementos están enlazados entre sí en la memoria o contienen enlaces a objetos comunes. El efecto de esta división es crear más producciones pero menos exigentes, por tanto de aplicación más general, de lo que sería una única producción. Así, cada grupo identificado da lugar a una producción según el siguiente esquema:

1) Se localizan las trazas que contengan alguno de los elementos del grupo anotado como elemento de acción de la traza. Dichos elementos serán las acciones de la nueva producción.
2) Los elementos de condición de las trazas halladas en 1) se dividen, a su vez, en dos subconjuntos:

- Elementos que ya existían antes de generarse el subobjetivo. La existencia de estos elementos pasa a exigirse como condiciones de la nueva producción.
- Elementos generados durante el proceso del subobjetivo. Estos se analizan recursivamente, con objeto de hallar nuevos elementos de condición que dieron lugar a su creación.

Como se desprende de lo anterior, al análisis realizado (desde los resultados obtenidos al procesar un objetivo) incorpora solo aquellos elementos de la memoria de trabajo que han sido relevantes en el proceso, desechando aquellos que únicamente han formado parte de caminos de búsqueda infructuosos.

Como paso final en la generación automática de producciones se realizan dos optimizaciones:

1) Reordenación de las producciones, buscando la eficiencia del proceso de "pattern-matching" [Smith, Genesereth 85].
2) Para las producciones aprendidas para implementar un operador complejo, creación de una producción separada de estructura simple (un único par condición-acción) para copiar cada estructura común entre el estado de partida y el de llegada. Todas estas producciones se disparan en paralelo (no importa el orden) y eliminan la explosión combinatoria que resultaría de considerar los pares condición-acción (todos juntos) de forma integrada, en las producciones aprendidas originales.

2.2.2.2. Discusión.

En resumen, de acuerdo con la descripción anterior, el concepto de sistemas tipo SOAR ofrece la posibilidad de formular modelos para el tratamiento de problemas en base a entidades cognitivas apoyadas en [Cuena 91]:

1) Un formato para definir espacios problema, donde se plantean la clase de problemas a resolver y el conocimiento para resolverlos.
2) Un conjunto de procedimientos (todos ellos sistemas de reglas de producción) para inferir cambios a nivel básico e inferir preferencias a nivel del control.
3) Un mecanismos de aprendizaje automático ("Chunking").
4) Una base general de producciones para control general.

En base a esto, un modelo SOAR se describe por un conjunto de instancias del concepto genérico de espacios problema (ver figura 2e [Cuena 91]). Aunque en la publicación original de SOAR [Laird et al 87] se planteaba la memoria de producciones de una forma global a nivel de implementación, desde el punto de vista conceptual de construcción de modelos, se considera más adecuada la organización antedescrita de la memoria de producción estructurada en producciones locales a espacios problema y generales para control. Asimismo, en la figura 2e se

incluye como parte del modelo la base de conocimiento de control ya que, aunque se planteó como interna a SOAR, es concebible dentro del concepto de arquitectura que se formule ad hoc para una clase de problemas.

SOAR es, pues, un ejemplo paradigmático de arquitectura modular en el que se tienen diferentes inteligencias especializadas en subproblemas del problema global, más una inteligencia general de control (basada en el mecanismo de "subgoaling"). Cada una de esas inteligencias es del mismo tipo y se implementa de la misma forma, en base a una memoria de producciones sobre la que se actúa teniendo en cuenta la pila de contextos. Desde este punto de vista la arquitectura puede considerarse uniforme.

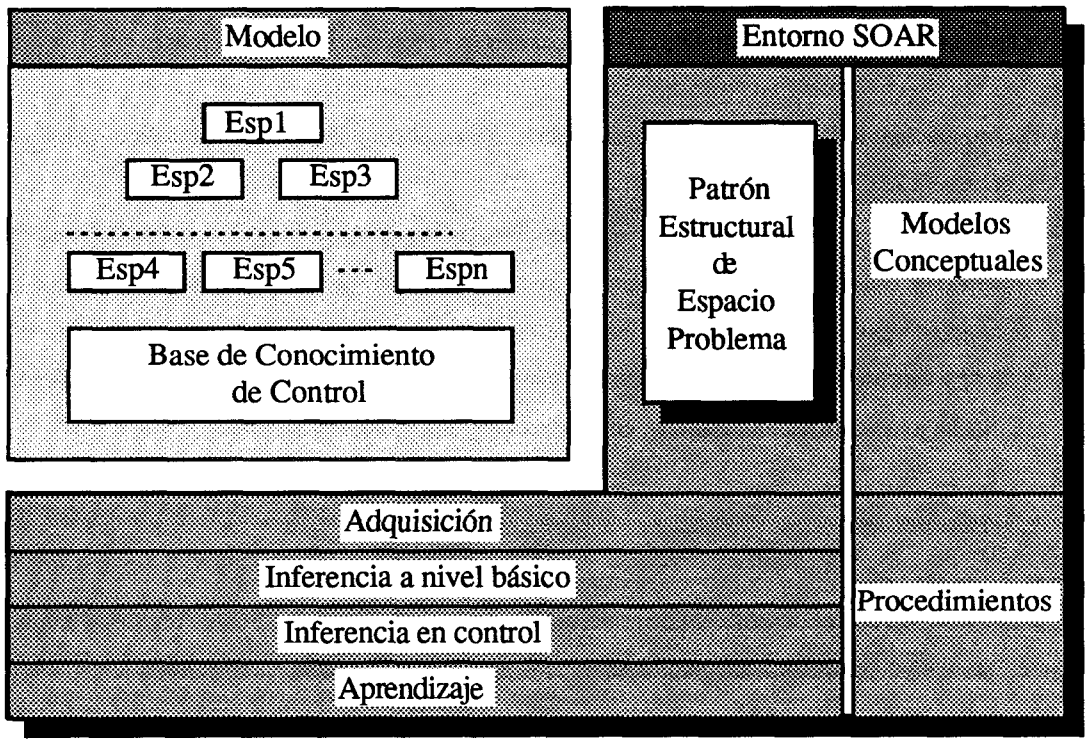


Figura 2e: Construcción de modelos con el entorno SOAR.

SOAR es un sistema cuyos mecanismos de inferencia son todos del mismo tipo (basados en reglas de producción). Cuando la tarea de resolver un problema o subproblema entra en una situación de "impasse" la tarea de control divide el problema mediante el mecanismo de "subgoaling". En este aspecto, el enfoque de SOAR, o al menos la implementación de Laird y Rosenbloom, es limitada respecto de, por ejemplo, el enfoque basado en Tareas Genéricas de Chandrasekaran, donde la estrategia para resolver un problema está más cerca de la forma en que un experto entiende ese proceso. El enfoque de Chandrasekaran está en la linea de construir resolvedores especializados de problemas, en contraposición a la idea de resolvedor universal, con representación estructurada del conocimiento, presente en SOAR. Bajo esta perspectiva, SOAR sigue instalado en la linea utópica de finales de los sesenta en que se consideraba la posibilidad de construir modelos útiles (siendo universales) de resolución de problemas.

En el momento actual, sin embargo, es necesario que la Inteligencia Artificial se encamine hacia la resolución de tipos de problemas por especialización no solo del conocimiento que se representa, sino de la estrategia para utilizarlo.

El comentario anterior debe entenderse como una crítica no tanto a la idea de la representación del conocimiento presente en SOAR, cuyo enfoque jerárquico por descomposición en espacios problema es válido y utilizable, como a la interpretación limitada de su concepto que se recoge en la implementación de [Laird et al 87]. De hecho, el mecanismo de manejo de preferencias de SOAR ha sido fuente de inspiración para establecer que tipos de criterios habían de tenerse en cuenta en las Tareas de Control desarrolladas en la tesis.

2.2.3. Arquitecturas basadas en Tareas Genéricas.

La lógica, los lenguajes basados en reglas y la representación en marcos han sido los tipos más populares de propuesta para la representación del conocimiento. Cada una de estas representaciones tiene una familia de mecanismos de inferencia capaces de operar sobre ellas. Cada tipo de representación, junto con su esquema de inferencia apropiado, definen para el utilizador una manera de formular modelos cognitivos y, a nivel del software, una arquitectura de implementación constituida por el conjunto de procedimientos y estructuras de información de soporte a la representación.

Las arquitecturas clásicas del tipo anterior permiten, a priori, la ejecución de cualquier tipo de tarea; sin embargo, su mayor critica estriba en que no ofrecen formatos de definición del conocimiento, tanto de dominio como de control, que sean naturales para el tipo de tarea que se ejecuta. Por ejemplo, cabe pensar, intuitivamente, que el razonamiento en diagnostico requiere tipos de conocimiento y estrategias de control similares en diferentes dominios. De igual forma se puede razonar para problemas de interpretación, planificación, predicción, etc. No obstante, también parece razonable pensar que cada uno de los tipos de problemas mencionados requieren estructuras de representación y estrategias generalmente diferentes.

A pesar del comentario anterior, cuando se examinan los esquemas de representación que se utilizan normalmente en el diseño de sistemas basados en el conocimiento, la representación del conocimiento y las estrategias básicas no tienen en cuenta las diferencias y peculiaridades de cada tipo de problema. Por ejemplo, en un problema de diagnostico cabe hablar en términos de clases de averías, estrategias de confirmación y rechazo, etc, mientras que en un problema de planificación se habla, probablemente, de jerarquía de componentes, planes, acciones, etc.

Idealmente, pues, se desearía representar el conocimiento con el vocabulario adecuado para el tipo de problema a resolver. Por el contrario, los lenguajes a que han dado lugar los sistemas expertos clásicos implementados han

uniformizado la representación, naufragando generalmente en las peculiaridades de cada clase de problema. Como consecuencia, la construcción actual de sistemas basados en el conocimiento se plantea, al menos en el ámbito industrial, enfocada a la implementación; es decir, se asume que el conocimiento se puede traducir más o menos directamente a estructuras básicas como reglas, marcos, etc, sin representar fielmente el proceso real de resolución que realiza el experto humano.

La naturaleza del conocimiento necesario para resolver cada tipo de problema así como la propia estrategia de resolución son aspectos, pues, a tener en cuenta con independencia de los mecanismos que, finalmente, se apliquen para su programación. El nivel del conocimiento de [Newell 82] cumple este papel, centrando el análisis del conocimiento en establecer las características funcionales de la tarea abstracta (el agente inteligente) y el conocimiento requerido para realizarla. Si bien la representación a este nivel no tiene por qué ser completa, el nivel del conocimiento sirve como guía razonable para estructurar tanto la representación a nivel simbólico como, finalmente, la implementación.

En esta línea, durante los ochenta han aparecido diferentes propuestas para abordar el análisis del conocimiento, dando lugar a diferentes formas de encapsular conocimiento y estrategia. Entre ellas cabe destacar:

1) El enfoque orientado al problema debido a McDermott.
2) El enfoque orientado al método de resolución debido a Chandrasekaran.

En los apartados siguientes se discuten cada una de estas propuestas.

2.2.3.1. El enfoque orientado al problema.

Bajo este enfoque se pretende definir una taxonomía de métodos de resolución de problemas, identificando la clase de problema para el que cada método resulta más adecuado [McDermott 89].

La resolución de problemas se presenta como el proceso de identificación, selección e implementación de una secuencia de acciones para realizar una tarea dentro de un dominio específico. Un método de resolución debe proporcionar el medio para:

1) Identificar acciones posibles, utilizando conocimiento dependiente del dominio.
2) Seleccionar entre acciones candidatas, utilizando mecanismos de control propios del método de resolución.
3) Ejecutar la acción seleccionada.

Nótese como esta caracterización de un método de resolución se basa fundamentalmente en el papel que el conocimiento juega en el proceso de resolución, en particular el conocimiento de control. Este planteamiento, que da lugar a los métodos comúnmente denominados "role-limiting", contrasta con el de los métodos generales o débiles ("weak methods" [Laird, Newell 83] [Laird et.al 87]), aplicables potencialmente a muchos tipos de tareas, cuya generalidad se basa en que el conocimiento de control necesario y específico de la tarea se proporciona antes de aplicar el método, sin formar parte de este. Por ejemplo, el método "Hill Climbing" necesita como conocimiento de control una función de evaluación que determine si la acción candidata acerca o aleja al sistema de la solución.

Por el contrario, los métodos del tipo "role limiting" se caracterizan por tener predeterminada tanto la forma de adquisición del conocimiento como su representación, estando el conocimiento de tipo estratégico incorporado en el propio método.

El sistema MYCIN [Buchanan et.al 84], es el primero en introducir un método de este tipo. Posteriormente, [Clancey 85] generaliza la estructura inferencial utilizada en el MYCIN dando lugar al método de Clasificación Heurística. Este método es similar a los métodos débiles en que su estructura de control es muy simple, pero el conocimiento de control relacionado con la tarea ya está incluido en la definición del método, lo que simplifica su utilización. La estructura inferencial resultante divide la tarea de razonamiento en tres pasos de inferencia: Abstracción de los datos de entrada, Equiparación de los datos con una clase de soluciones y Refino de la solución. Los pasos de inferencia se encadenan en secuencia, bajo dos estrategias básicas posibles (de los datos a la solución o viceversa) según se muestra en la figura 2f.

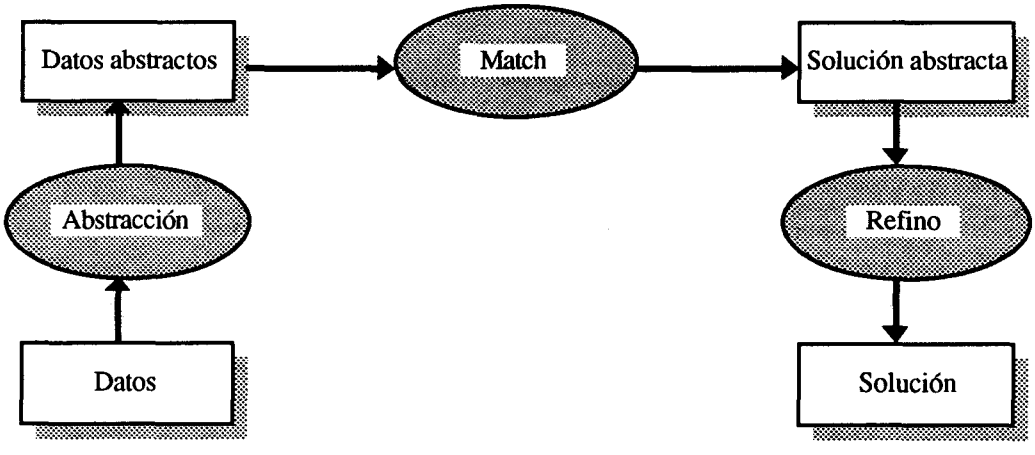


Figura 2f: Clasificación Heurística.

Realizar el análisis del conocimiento con ayuda de estructuras de este tipo tiene ventajas indudables. En primer lugar, hace explicita una cierta estructuración del conocimiento que, de otra forma, termina realizándose de forma artificiosa mediante las estructuras básicas de representación. En segundo lugar, el análisis se realiza, al menos a

priori, de forma independiente de los esquemas que luego se utilicen para implementar la representación y ayudando, de hecho, a realizar una elección adecuada -con más criterio- de dichos esquemas. Finalmente, y quizá lo más importante, al describir el patrón de inferencias a realizar, la estructura aporta al menos una pauta para abordar la adquisición del conocimiento.

El principal problema con este tipo de estructura es que casi cualquier sistema puede ser analizado en base a ella. Aunque aporta un cierto grado de estructuración al proceso inferencial global, basado en la idea de que el razonamiento se hace mejor a niveles altos de abstracción, no sugiere ningún principio de estructuración del conocimiento del dominio, precisamente por su grado de generalidad, alejándose por tanto de la forma en que un experto puede entender la tarea de resolver un problema.

Más en la linea de identificar arquitecturas acordes con las formas de entender, [McDermott 89] establece una taxonomía de métodos de resolución, del tipo "role limiting", con objeto de llegar a construir herramientas que permitan realizar la adquisición y representación del conocimiento, para cada tipo de problema, de forma automática. Este planteamiento deja a un lado aquellos problemas que no son fácilmente resolubles por métodos "role limiting"; es decir, en aquellos casos en los que sea necesario incorporar conocimiento de control en la base de conocimiento (por ejemplo conocimiento de preferencias entre acciones candidatas del tipo utilizado en SOAR), por estar muchas veces íntimamente ligado al dominio de aplicación.

La taxonomía de McDermott define, para cada método identificado, las características que deben poseer los problemas para los que el método es adecuado, así como la estrategia de control a seguir. Los ejemplos más significativos de la taxonomía son:

1) *Cubrir y Diferenciar*, que es una forma de clasificación heurística adecuada para problemas de diagnostico. Las características que debe poseer un problema para que el método sea aplicable se resumen de la siguiente forma:
   - Existen un conjunto de síntomas o anormalidades que deben ser explicadas.
   - Es posible establecer una enumeración de explicaciones candidatas (causas) que cubren el conjunto de síntomas.
   - Existe alguna información que permite diferenciar entre varias explicaciones alternativas para un mismo síntoma.
   - Normalmente solo es aplicable una explicación para cada síntoma.

   La estrategia de control se expresa en tres grandes pasos:
   - Entrada de síntomas o anormalidades.
   - Identificación de hipótesis de explicación candidatas (cubrir)

- Para cada síntoma, si hay más de una hipótesis de explicación, utilizar el conocimiento que permite hacer selección entre causas (diferenciar).

2) *Cubrir y Diferenciar incluyendo Razonamiento Cualitativo*, apropiado para problemas de diagnostico sobre sistemas físicos. Este método pone en combinación las técnicas de razonamiento cualitativo (en particular los paradigmas de DeKleer o Kuipers comentados en el capítulo tres) con el método anterior. Las características que debe poseer un problema para que el método sea aplicable se resumen de la siguiente forma:
   - Existen un conjunto de síntomas o anormalidades que deben ser explicadas.
   - Es posible proporcionar un modelo de funcionamiento normal del sistema, caracterizado en términos de restricciones cualitativas que ligan variables de estado descriptivas de cada componente del sistema.
   - Los posibles fallos se pueden expresar como perturbaciones que afectan al funcionamiento de un componente o varios que, de esta forma, responden a diferentes leyes de comportamiento de las establecidas.
   - Existe información para discriminar entre fallos alternativos que explican un mismo síntoma.
   - Normalmente, solo es aplicable una explicación para cada síntoma.

La estrategia de control se expresa en tres grandes pasos:
   - Dada una variable cuyo valor no es normal en un momento dado, establecer el conjunto de restricciones que influyen, directa o indirectamente, en ella. Esto se hace remontando a través de las ecuaciones en las que la variable es dependiente hasta encontrar ecuaciones cuyas variables independientes son entradas al sistema (son medidas).
   - Verificar la consistencia del conjunto de restricciones anterior, identificando las restricciones que, potencialmente, justifican la inconsistencia.
   - Resolver conflictos, seleccionando una restricción (por tanto un componente del sistema) que explica el funcionamiento anormal, posiblemente utilizando conocimiento de tipo heurístico.

3) *Proponer y Revisar*, apropiado para problemas de diseño. Las características que debe poseer un problema para que el método sea aplicable se resumen de la siguiente forma:
   - Es posible especificar procedimientos para determinar una configuración de diseño inicial.
   - Para cada restricción de diseño, deben existir indicaciones, en forma de acciones a seguir para modificar el diseño, si la restricción no se cumple.

La estrategia de control se expresa en cuatro grandes pasos:
   - Se aceptan un conjunto de especificaciones iniciales.
   - Se añaden elementos al diseño inicial, comprobando que se cumplen las restricciones.
   - Para cada restricción que no se satisface, se busca el cambio de menor coste que permite satisfacer la restricción. Dicho cambio se aplica.
   - El proceso continúa hasta que el diseño está completo y no se viola ninguna restricción.

El enfoque centrado en tipos de problemas tiene como ventaja importante el aportar criterio para clasificar un problema dado. De esta forma, se ponen de relieve rápidamente las necesidades de conocimiento sobre el dominio que han de satisfacerse, a la par que se introduce una cierta sistemática en el proceso de adquisición. Esto puede conducir a metodologías de adquisición del conocimiento con un alto grado de formalización como es el caso de la metodología KADS [Breuker, Wielinga 84b, 85, 86] [Hitchman et.al 89]. El objetivo final, aunque lejano, puede ser la generación de herramientas que, implementando dichas metodologías, sean directamente utilizables por los expertos en el dominio de aplicación.

Por otra parte, la existencia de métodos de resolución por clases de problemas puede servir de base para generar explicaciones de mayor calidad; es decir, que van mas allá de la pura revisión de las estructuras de representación utilizadas durante el razonamiento. Por ejemplo, es posible explicar en términos de la fase del método que se está ejecutando y del conocimiento del dominio utilizado.

La principal desventaja de este enfoque estriba en que, si bien aporta criterio para clasificar problemas y para estructurar la estrategia general de control, no hace lo propio con el conocimiento del dominio. Análogamente a lo ya comentado para la Clasificación Heurística, los métodos de resolución no sugieren como debe ser el modelo del conocimiento del dominio. En problemas en los que se requiere una representación profunda de dicho conocimiento, por ejemplo para hacer razonamiento cualitativo sobre sistemas físicos, no basta con identificar adecuadamente el tipo de problema a resolver sino que es necesario, además, tener criterio para elegir el paradigma de modelización (ontología) más adecuado.

Desafortunadamente, este ultimo aspecto no depende tanto del tipo de problema como de la clase de dominio de que se trate. Por ejemplo, en hidráulica puede ser muy adecuado utilizar un enfoque centrado en procesos [Forbus 84, 89] mientras que para diagnostico de fallos en circuitos eléctricos el enfoque centrado en componentes [DeKleer, Brown 84a,84b] será probablemente más útil.

2.2.3.2. El enfoque orientado al método.

A finales de los setenta, paralelamente a la aparición de la arquitectura derivada del MYCIN, Chandrasekaran inicia su propia linea de investigación, cuyo origen es la propuesta de la Clasificación como una tarea básica a ejecutar en la resolución de problemas de diagnóstico.

Esta propuesta culmina con el desarrollo del sistema MDX [Chandrasekaran, Mittal 82]. MDX es un sistema experto en diagnóstico médico en el que se intenta definir la estructura del conocimiento acorde con la forma en

que los expertos humanos abordan la tarea de diagnosticar en ese dominio. Por su parte, el conocimiento sobre el dominio se organiza en una jerarquía clasificativa de la siguiente forma: el conocimiento se descompone en subestructuras, cada una de las cuales está especializada en resolver un tipo de problema. Cada subestructura se descompone, a su vez, en pequeñas fuentes de conocimiento organizadas como una jerarquía de especialistas que difieren uno del otro no en el método de resolución, sino en el contenido conceptual de su conocimiento. Por ejemplo, un especialista puede entender de enfermedades del corazón, otro en enfermedades del hígado. El esquema se completa con la incorporación, para cada subestructura, de un módulo de interpretación de datos observables, con su propia estructura del conocimiento.

El sistema MDX pone de manifiesto la utilidad de descomponer la tarea global de diagnóstico en subtareas (Clasificación e Interpretación), cada uno con su propia estructura del conocimiento y sus patrones de inferencia, dando lugar a una forma de modularización en la que, por cada módulo, se sugieren tanto las estrategias de control como el principio de organización del conocimiento sobre el dominio, entendiendo que ambos aspectos están íntimamente ligados.

El concepto de la representación utilizado en el sistema MDX es precursor del enfoque basado en Tareas Genéricas. Aunque originalmente se presenta como otra propuesta de taxonomía de tipos de resolución de problemas, en la linea de McDermott, el principio de modularización (mediante la tarea genérica) es sin embargo distinto. Efectivamente, el enfoque de tareas genéricas propone la construcción de sistemas basados en el conocimiento mediante bloques, definidos como abstracciones funcionales, cada uno de los cuales entiende de un aspecto parcial del problema. Por ejemplo la Tarea Genérica Clasificativa (como se verá más adelante) es un bloque funcional que entiende de como resolver una parte del problema de diagnóstico, no el problema completo.

Cada tarea genérica utiliza una estructura de representación y una estrategia de control propios y, conceptualmente, cercanos al conocimiento del dominio. Se incorpora la idea, por tanto, de que el tipo de problema (o del subproblema) a resolver puede determinar el modo de resolverlo, considerandose que el conocimiento y su utilización son dos aspectos íntimamente ligados para una clase dada de problemas, aunque se mantengan separados desde el punto de vista de la implementación. Forzar una separación del conocimiento y el proceso que lo utiliza puede dar, a menudo, una falsa idea de generalidad ya que las arquitecturas de soporte para mecanismos de control de tipo uniforme suprimen las diferencias que, de hecho, existen en control e inferencia entre diferentes tipos de tareas.

Básicamente, una tarea genérica se caracteriza por:

1) El tipo de información que la tarea necesita como entrada y el tipo de información que la tarea produce como resultado de su ejecución.

2) Un principio de organización del conocimiento (por ejemplo, una jerarquía de especialistas según la relación ES PARTE DE).
3) La estrategia general de control que la tarea utiliza.

Entre las distintas tareas identificadas, cabe destacar las siguientes:

1) Clasificación jerárquica. Consiste básicamente en encontrar el nodo o nodos, dentro de una jerarquía, aplicables a la situación que se analiza.
2) Selección de un plan y refino. Consiste en generar modelos de diseño de un objeto, acordes con unas especificaciones, utilizando una forma de planificación jerárquica.
3) Equiparación de hipótesis ("Hipothesis matching"). Consiste en identificar una hipótesis a partir de una situación de entrada.
4) Ensamblaje de hipótesis. Consiste en construir una hipótesis compuesta que puede explicar un conjunto de datos de entrada.

A continuación se describen por separado las dos primeras tareas entre las enumeradas, por ser las más ampliamente utilizadas.

Clasificación jerárquica.

El concepto de clasificación jerárquica parte de la base de que la clasificación es un tipo de razonamiento típicamente utilizado por las personas. La clasificación jerárquica es una forma particular de clasificación que, como tarea genérica, posee las características siguientes:

1) Necesita, como entrada, una descripción del problema a resolver y produce, como salida, el conjunto de hipótesis aplicables a los datos de entrada.
2) El conocimiento que se requiere para efectuar la tarea consiste en una lista de hipótesis a considerar y cuyo principio de organización es una jerarquía en la que los nodos hijos de un nodo dado representa subhipótesis del nodo padre; es decir, son hipótesis más especificas. Cada nodo en la jerarquía contiene el conocimiento necesario para confirmar o rechazar la hipótesis que representa, pudiendo considerarse, a todos los efectos, como un especialista en su pequeño dominio. Este concepto de especialización es, de hecho, recursivo; es decir, la funcionalidad de un especialista puede ser resuelta mediante otra tarea.

La figura 2g muestra un ejemplo de jerarquía clasificativa aplicada al diagnóstico de problemas de fuel en motores de automóviles.

La estrategia de control de la clasificación jerárquica consiste básicamente en atravesar la jerarquía eficientemente. Para ello se utiliza un proceso de confirmación y refino: un especialista confirma su hipótesis y se refina a sí mismo activando subespecialistas mas específicos, mientras que un especialista que rechaza su hipótesis impide la búsqueda en el subárbol por debajo de él. El proceso se repite hasta que no es posible refinar más, bien porque se alcanzan las hojas de la jerarquía, bien por que todos los especialistas rechazan su hipótesis.

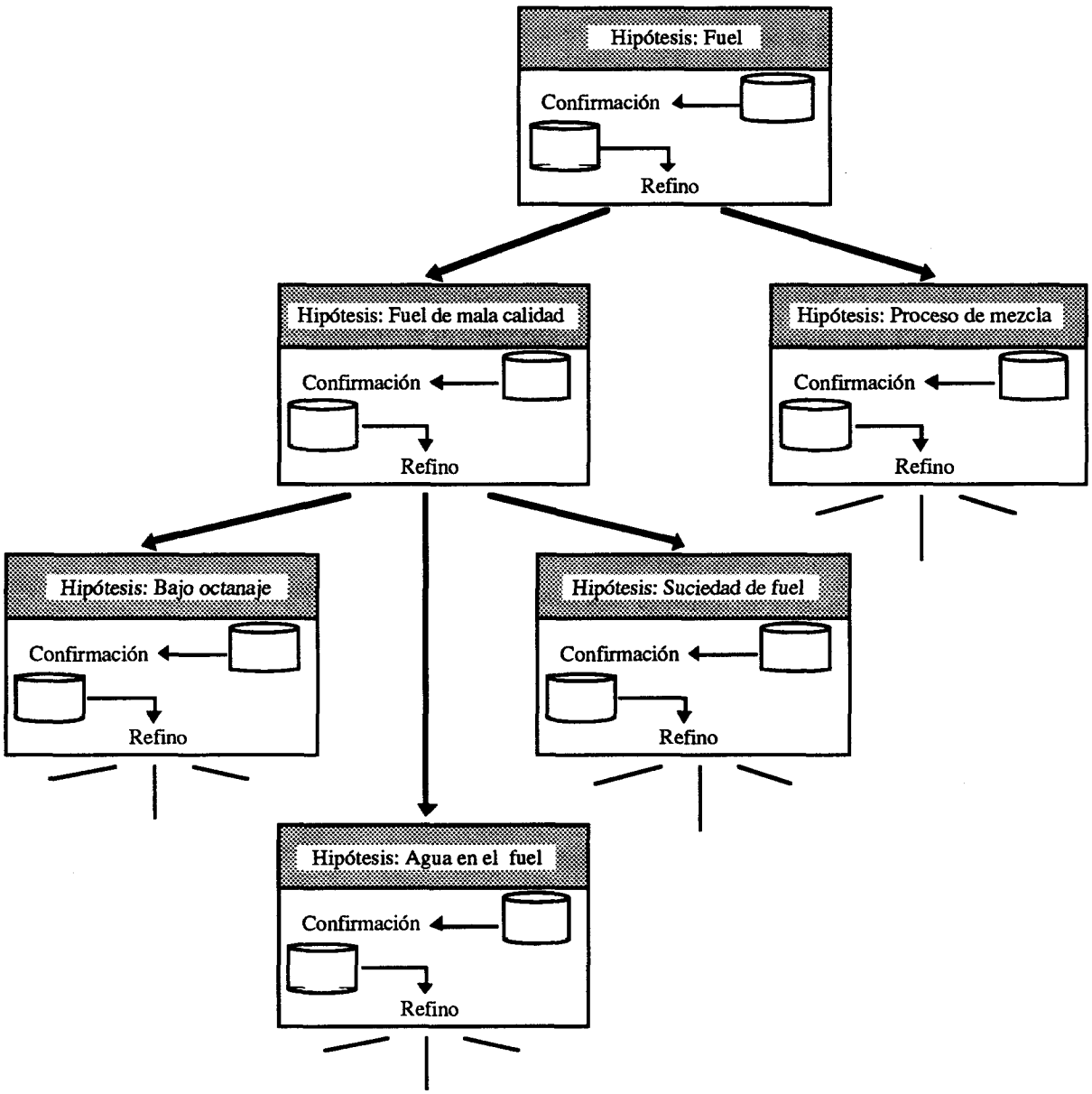


Figura 2g: Organización del conocimiento según una jerarquía clasificativa.

Un aspecto importante de la jerarquía clasificativa es su forma de integrar el conocimiento y el mecanismo de inferencia; es decir, el principio de organización del conocimiento es, al mismo tiempo, el hilo conductor de la

estrategia de resolución del problema. Realmente, el mecanismo de inferencia está embebido en cada nodo especialista, que tiene capacidad para decidir la relevancia de su hipótesis particular. De esta forma, el conocimiento está organizado de acuerdo con el problema que se necesita resolver.

La tarea clasificativa puede servir como bloque funcional utilizable como componente en arquitecturas cognitivas más complejas, involucrando razonamiento multinivel. Por ejemplo, la resolución de un problema de diagnóstico puede verse como una tarea compleja cuyo objetivo es explicar un conjunto de observaciones en un sistema. Las explicaciones se producen en términos de las averías que pueden haber actuado como causa de las observaciones. El resultado debe ser un conjunto de hipótesis de avería que expliquen todas las observaciones, sin incluir hipótesis innecesarias y teniendo en cuenta las posibles interacciones entre hipótesis.

De manera muy similar al esquema de métodos de resolución de McDermott, los problemas sobre los cuales es aplicable la tarea anterior deben satisfacer una serie de propiedades:

1) El conocimiento está disponible como una jerarquía de averías articulada mediante una relación del tipo clase-subclase, ES PARTE DE, etc.
2) Dada una observación, solo un conjunto pequeño de averías pueden estar implicadas; es decir, las interacciones entre averías con respecto a una observación son limitadas.

El conocimiento en dominios que cumplan las propiedades anteriores hace posible descomponer la resolución del problema en dos módulos tarea: una tarea clasificativa, que utiliza la jerarquía para seleccionar un conjunto de posibles hipótesis de explicación, y una tarea de refino que, probablemente utilizando razonamiento de tipo abductivo, reduce el conjunto de hipótesis generando una combinación de las mismas que explique de forma coherente las observaciones de entrada.

Notese como, si bien las propiedades anteriores son similares a las exigidas para el método *Cubrir y Diferenciar*, son más exigentes al forzar una forma determinada de organizar el conocimiento lo cual, al mismo tiempo, es su principal ventaja ya que suministra una pauta para su adquisición.

Selección de Planes y Refino.

De forma abstracta, el diseño puede definirse como la especificación de una estructura de componentes relacionados que satisfacen un conjunto de restricciones. Aún siendo una actividad compleja, existen casos de diseño con características restrictivas:

1) El problema inicial de diseño puede descomponerse en un número de pequeños subproblemas los cuales, a su vez, admiten descomposición, etc.
2) Para cada subproblema existen métodos conocidos, planes de diseño, capaces de producir diseños satisfactorios.

Los problemas con las características citadas puede parecer simples pero aún requieren, para su solución, sistemas basados en el conocimiento: es necesario seleccionar adecuadamente entre planes alternativos y, ante la posibilidad de fallos, realizar un complejo "backtracking".

La Selección de Planes y Refino es una tarea genérica utilizable para resolver este tipo de problemas. El criterio requerido para estructurar el conocimiento vuelve a ser una jerarquía de especialistas (en diseño) en la que en los niveles más altos se sitúan los especialistas en aspectos más generales del objeto a diseñar, en tanto que los niveles más bajos se ven ocupados por especialistas en componentes más específicos. Para la construcción de un modelo del conocimiento para diseño, el ingeniero dispone de las siguientes primitivas:

1) Especialistas. Un especialista es un agente para el diseño de una parte del objeto global (objetivo del diseño). Su nivel de responsabilidad en el proceso de diseño depende del nivel de la jerarquía en el que se sitúe. Obviamente, el especialista situado en la raíz es responsable del diseño completo.
   Para realizar su función, un especialista utiliza conocimiento propio o puede requerir los servicios de otros especialistas. El conocimiento propio se estructura en:
   - Una colección de planes. Cada plan consiste en una serie de llamadas a subespecialistas, que refinarán el diseño, y a tareas, representando métodos alternativos para diseñar la parte del objeto en la que el especialista entiende.
   - Un conjunto de restricciones para verificar si se satisfacen los requerimientos y, por tanto, el especialista ha operado con éxito.
2) Tareas. Una tarea es un agente de diseño expresado como una secuencia de acciones con restricciones de ejecución. Es responsable de la coherencia lógica, estructural o funcional de una parte de un componente (por ejemplo, decidir el diámetro de un agujero para un tornillo).
3) Acciones. Un acción es un agente de diseño capaz de tomar una decisión simple, generalmente afectando a un único atributo del diseño. Por ejemplo, decidir sobre el material más adecuado para un componente simple o su longitud, etc.

La figura 2h muestra la estructuración del conocimiento para diseño.

La estrategia básica de resolución es como sigue:

1) Se obtienen los requerimientos del diseño (especificaciones).
2) El diseño comienza en la raíz de la jerarquía y progresa hacia las hojas de la misma. Un especialista comienza recibiendo una petición de diseño desde su superior en la jerarquía, selecciona un plan acorde con la petición y lo ejecuta, produciendo llamadas a especialistas subordinados con peticiones de diseño de subcomponentes.
3) Cuando las restricciones de diseño establecidas en una pieza de conocimiento no se satisfacen se produce un mensaje de fallo al agente superior (tarea, plan o especialista) dando lugar a un punto de retroceso. Si el agente superior es un especialista, seleccionará otro plan alternativo, prosiguiendo el proceso.

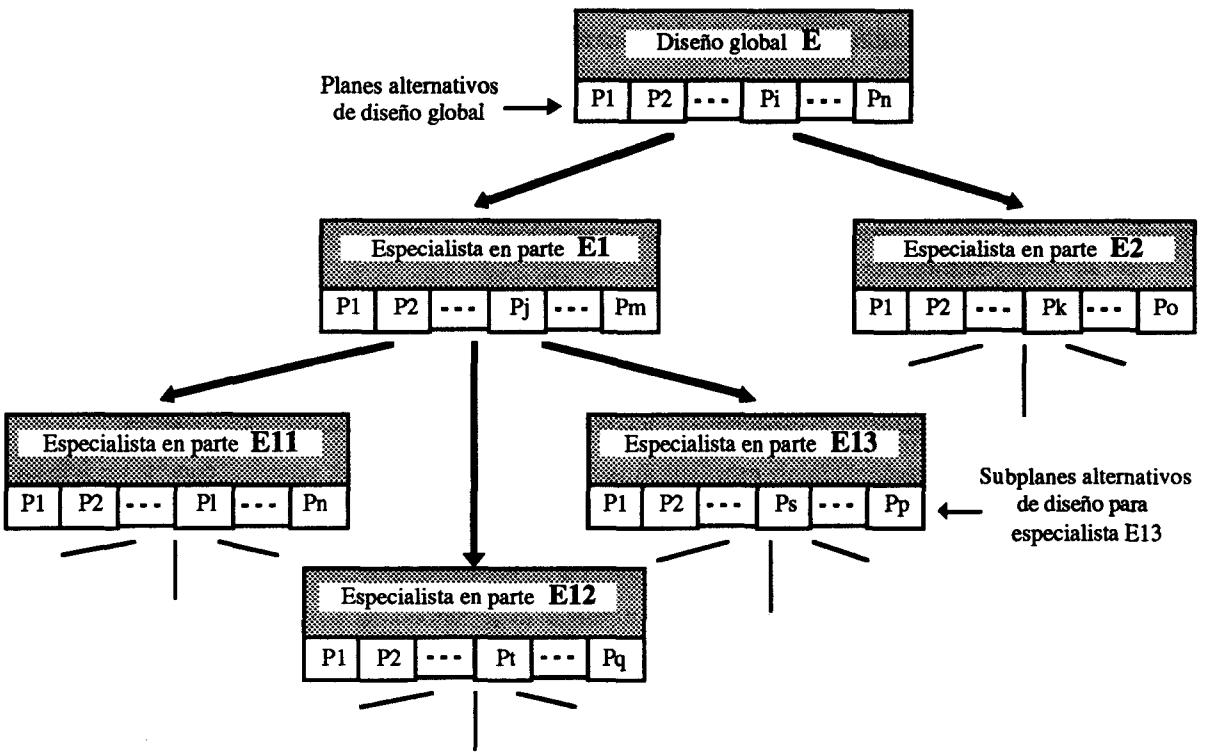


Figura 2h: Estructura del conocimiento para diseño.

El enfoque anterior aborda el proceso de diseño de forma estructurada, donde la estructuración se produce por vía cognitiva [Cuena 91], superando el esquema clásico que utilizaba R1 [McDermott 82] cuya organización se basaba en el uso de contextos de razonamiento rígidamente conectados mediante reglas prefijadas.

El concepto estructurado de diseño puede generalizarse mediante la inclusión de tareas de planificación de tipo básico, definiendo especialistas cuyo conocimiento se represente mediante conjuntos de operadores que recojan las acciones posibles dentro de un contexto. De esta forma, ese tipo de especialista realiza razonamiento en planificación en lugar de razonar de forma clasificativa. Por otra parte, el manejo del catálogo de planes alternativos podría también sofisticarse en linea con la técnica "case based reasoning" [Kolodner 87]

[Chandrasekaran 88], en la que se elige un plan aunque no encaje totalmente en el caso a resolver y se hace un razonamiento de adaptación para producir un plan más adecuado.

2.2.3.3. Discusión.

La concepción de tareas genéricas como bloques funcionales permite la identificación de tipos de razonamiento que, siendo genéricos, pueden utilizarse como componentes de tareas de razonamiento mas complejas, dando lugar a arquitecturas cognitivas formadas por composición de esos bloques. La estructura de tarea viene a ser una versión análoga, aunque más general, que los espacios problema de SOAR, permitiendo individualizar áreas de inferencia y representación por clases de problemas o subproblemas. Esto puede conducir, teóricamente, a la construcción de bibliotecas de tareas genéricas y su inclusión en un entorno para representación del conocimiento sobre una clase de problemas, así como a definir una sistemática para realizar la tarea de analizar el conocimiento necesario para resolver un problema, según un proceso "top-down". Un modelo, dentro de un entorno de estas características, se crearía mediante instancias particularizadas de los tipos de tareas definidos, según el esquema mostrado en la figura 2i. El modelo, más la estrategia global, constituyen la tarea de más alto nivel, capaz de resolver problemas en el dominio para el que se diseña.

Esto tiene gran analogía con el enfoque de Breuker y Wielinga en la metodología KADS. Aunque en [Chandrasekaran et al. 92] se afirma que la metodología KADS propone un esquema "bottom-up" para realizar el análisis del conocimiento, ello es así quizá solo parcialmente. KADS sugiere una aproximación ascendente al análisis solo si, en el catálogo de tareas de la metodología, no existe una tarea -o conjunto de tareas, que se ajusten adecuadamente al tipo de problema a resolver. En caso contrario, el método es claramente "top-down".

De todas formas, carece de sentido establecer comparaciones directas entre ambos enfoques, porque son de naturaleza relativamente distinta. Chandrasekaran no aporta una sistemática para el análisis sino una estructuración de tipo cognitivo que puede facilitar su realización. Aunque en [Chandrasekaran 87], [Chandrasekaran et al. 92] se habla de "la metodología derivada (detrás) del uso de tareas genéricas", la realidad es que no se define un método paso a paso para conducir la tarea del análisis del conocimiento. Por el contrario, Breuker y Wielinga si definen metodología, y no solamente para conducir el análisis del conocimiento (llamado análisis interno en KADS) sino cubriendo el ciclo completo de construcción de un sistema basado en el conocimiento utilizando Tareas Genéricas.

El objetivo, por tanto, va en la linea de conseguir tipos abstractos de conocimiento (en analogía con los tipos abstractos de datos) particularizables posteriormente para cada dominio concreto de manera que, cuando se estudian tareas de la misma naturaleza genérica -para dominios diferentes, se identificarán el mismo tipo de modelos del dominio y el mismo tipo de métodos de inferencia.

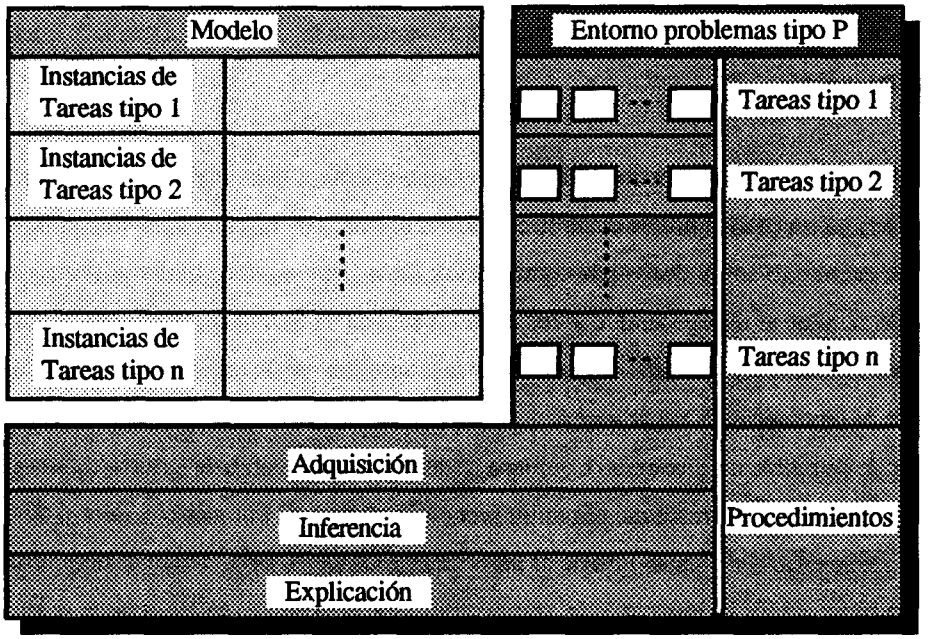


Figura 2i: Construcción de modelos con un entorno de Tareas Genéricas.

La última afirmación, sin embargo, parece excesivamente optimista, al menos al abordar la resolución de problemas relacionados con el mundo físico. En efecto, la organización del conocimiento, de cara a ejecutar la estrategia general de control, puede ser análoga en diferentes dominios; sin embargo, cuando se requiera construir modelos para efectuar razonamiento a partir de principios básicos, cada clase de dominio impondrá generalmente tanto la forma de modelización como el método de razonamiento, aún cuando el tipo de problema a resolver sea similar.

Por ejemplo, si se piensa en el problema del diagnostico en un sistema complejo, la Tarea Clasificativa puede ser una buena aproximación. Su utilización sugerirá organizar el conocimiento según una jerarquía de subtareas que "entienden" de cada componente del sistema, y cuyo procedimiento general de interpretación permitirá aislar la fuente del problema a cualquier nivel de detalle. Sin embargo, el conocimiento necesario para confirmar una hipótesis (es decir, el conocimiento propio de un especialista) puede ser de naturaleza muy diferente, dependiendo de la clase de dominio. En diagnostico médico posiblemente puede alcanzarse, para cada especialista, un modelo de tipo superficial que examine cuadros alternativos de síntomas para verificar la hipótesis. En diagnóstico de una planta de Acrilo-Nitrilo, por el contrario, será muy difícil obtener de un experto juicios sobre los aspectos finales del comportamiento de ningún componente de la planta, debiendose acudir, para cada componente, a modelos del comportamiento ejecutables por simulación cualitativa.

El enfoque de tareas genéricas pretende obtener una teoría atómica del uso del conocimiento, de manera que sea posible alcanzar procesos de resolución complejos como interacción de varios "átomos" (tareas). Un problema abierto es establecer formas de interacción entre tareas; es decir, como decidir el control entre tareas. Inicialmente, se plantea un control de tipo prefijado, donde cada proceso llama explícitamente a aquellos otros que necesita. Esto implica una pérdida de flexibilidad. Además, el control predeterminado puede ser claramente insuficiente si ese control es muy complejo; por ejemplo, si depende de la naturaleza de las tareas involucradas o de la estrategia global como es el caso de la simulación de sistemas físicos. [Chandrasekaran 87] plantea una forma alternativa de control de tipo oportunista, bajo el que un proceso que necesite una información envía un mensaje de difusión y queda a la espera de que algún otro proceso lo acepte y responda.

Esta forma de control facilitaría, a priori, la posibilidad de que pudieran añadirse nuevas piezas de conocimiento no preestablecidas durante el diseño del sistema. Sin embargo, un control distribuido de este tipo se acercaría a las estrategias de tipo oportunista propias de las arquitecturas de pizarra, jugando las tareas el papel de fuentes de conocimiento. Este planteamiento caería, por tanto, en los mismos defectos reseñados para las arquitecturas de pizarra en cuanto que no recogen fielmente el proceso de resolución de problemas que efectúan los expertos humanos, aspecto criticado en la literatura de tareas genéricas.

La tesis plantea una forma de control explícito entre tareas, definiendo módulos tarea (las TGEs) para realizar labores de comunicación entre tareas, así como una Tarea General de Control que permite representar explícitamente conocimiento sobre estrategia global, en base a redes pseudo-arboriformes de objetivos. Se intenta alcanzar un esquema de formulación del conocimiento de control capaz de recoger las peculiaridades de una clase de dominio y que, al mismo tiempo, aporte un vocabulario natural para el diseñador experto en el dominio de aplicación.

3. RAZONAMIENTO SOBRE EL MUNDO FISICO.

Uno de los campos de investigación más interesantes en la linea de obtener representaciones del conocimientos profundo es el de la Simulación Cualitativa de sistemas físicos. Hayes, con su manifiesto sobre física cualitativa [Hayes 79] y en su revisión [Hayes 85] lanzó un doble reto:

1) La Inteligencia Artificial debía concentrar sus esfuerzo no solo en problemas tipo sencillos ("toy problems") sino ser capaz de abordar problemas en dominios complejos con necesidad de mucho conocimiento.
2) La Inteligencia Artificial debería ser capaz de producir una teoría cualitativa sobre el comportamiento del mundo físico, trasladable finalmente a una representación en computador, pero formalizada completamente de forma previa.

Derivadas de este impulso se han producido tres propuestas principales que, a su vez, han sido origen de otras propuestas posteriores:

1) El enfoque de Kuipers, centrado en restricciones.
2) El enfoque de DeKleer y Brown, centrado en componentes.
3) El enfoque de Forbus, centrado en procesos.

Estas propuestas no se presentan de forma excluyente. De hecho, es posible plantearse un entorno unificado que las utilice de manera selectiva y complementaria como se demuestra en [Bredeweg, Wielinga 88]. Su objetivo es común: obtener descripciones del comportamiento de un sistema físico en base a transiciones de estado, siendo el aspecto diferenciador las primitivas de modelización que se utilizan en cada caso, tanto para caracterizar el estado del sistema como su forma de evolucionar en el tiempo.

En los siguientes apartados se introducen los conceptos básicos de la física cualitativa para comentar brevemente, a continuación, cada una de estas propuestas.

3.1. FISICA CUALITATIVA.

La Física Cualitativa es la rama de la Inteligencia Artificial que estudia las formas de representación y razonamiento alrededor del mundo físico. Su principal objetivo es capturar tanto el conocimiento de sentido común que permite a las personas comprender, de forma intuitiva, la mayoría de los fenómenos físicos, como las leyes y principios básicos sobre los que descansan las (a menudo muy complicadas) expresiones cuantitativas utilizadas por los especialistas.

Quizá, una forma muy simple -pero muy efectiva, de justificar el estudio y uso de la física cualitativa es afirmar que cualquier persona, sin formación en ciencias Físicas o Matemáticas, sabe mucho acerca del mundo físico que le rodea, sin necesidad de acudir a complejos modelos con muchas ecuaciones y parámetros. Por ejemplo, conducen automóviles, juegan al billar, cocinan, etc; es decir, realizan actividades que involucran procesos físicos complejos.

Esto no quiere decir, naturalmente, que la realidad sea más simple que los modelos al uso que la explican, pero sí que su interpretación no requiere, en general, acudir al aparato matemático contenido en esos modelos.

Por otra parte, la construcción de los modelos clásicos, utilizados en la física tradicional para explicar de forma efectiva situaciones reales, presenta aspectos problemáticos. [Bonissone, Valavanis 85] y [Forbus 89] clasifican esos problemas en tres tipos:

1) Problemas de modelización; es decir, como establecer buenas correspondencias entre objetos del mundo real y abstracciones útiles. Las herramientas clásicas de modelización en física, las ecuaciones, casi nunca son suficientes para capturar la totalidad de un fenómeno. Por el contrario, una ecuación modeliza, en general, un aspecto parcial del fenómeno bajo consideración. Además, dicha ecuación no siempre existe.

   Por otra parte, la existencia de ecuaciones tampoco resuelve los problemas. Un conjunto de ecuaciones para modelizar una situación real (normalmente compleja) rara vez admite solución analítica. Siendo así, plantear la ejecución de un modelo conduce inevitablemente a la simulación numérica.
2) Problemas del proceso de razonamiento. La aplicación de modelos vía simulación numérica requiere conocer detalles del modelo, de sus condiciones de aplicación y de sus parámetros, con mayor precisión de la que generalmente se dispone. A la simulación de tipo numérico se le asocia tradicionalmente costes computacionales altos pero, probablemente, su verdadero coste que, frecuentemente, la hace no factible, es la necesidad de abrir espacios de búsqueda alrededor de conjuntos de parámetros que, a menudo, son conocidos con muy poca precisión.
3) Problemas de interpretación. La simulación numérica arroja como resultado respuestas muy precisas, asumidas como ciertas un conjunto de hipótesis. Aún cuando este tipo de simulación sea tecnológicamente factible, lo que debe admitirse como realidad creciente debido al aumento progresivo de potencia de cálculo en computadores, ello puede ser, a veces, todavía insuficiente ya que siempre será necesaria alguna suerte de interpretación de los resultados, normalmente en manos de las personas que gobiernan la ejecución del modelo. En general, un conjunto de valores numéricos asignados a una lista de variables de estado, como consecuencia de campañas de simulación, no es un resultado útil por sí mismo, pues resulta poco descriptivo del comportamiento.

> Por otro lado, la simulación numérica adolece de ser un proceso ciego; es decir, no consciente del conocimiento puesto en juego e implícito al modelo. En muchas situaciones puede ser preferible una respuesta rápida, descriptiva y justificada, aunque sea poco precisa, en términos de los comportamientos posibles, antes que una predicción muy precisa pero ofrecida demasiado tarde y con poca semántica.

Conviene aclarar, no obstante, que el objetivo de la física cualitativa no es desplazar a la física clásica, como tampoco lo es, para la simulación cualitativa, sustituir a la simulación numérica como técnica para ejecutar modelos en computador. De hecho, un modelo numérico bien calibrado, y con rendimiento suficiente, será preferible en muchas situaciones. Por el contrario, el objetivo debe ser, más bien, que ambos enfoques actúen de forma complementaria. No en vano, los físicos han tenido siempre, y de forma previa a la concepción de sus modelos matemáticos, teorías de sentido común acerca del mundo. La física cualitativa pretende definir modelos, ejecutables en computador, capaces de capturar esas teorías.

3.1.1. Elementos de la Física Cualitativa.

Un modelo de un sistema físico debe abarcar dos aspectos:

1) Una caracterización del estado del sistema.
2) Un concepto de la evolución (como se producen los cambios de estado) del sistema en el tiempo.

Un modelo numérico realiza la caracterización del estado mediante un conjunto de variables, típicamente elementos de la recta real $\Re$ variando de forma continua en el tiempo. El concepto de la evolución se establece mediante un conjunto de ecuaciones diferenciales cuya solución, obtenida de forma analítica o por interpretación numérica, define el comportamiento del objeto físico como una sucesión de estados en el tiempo.

Si uno de los objetivos de la física cualitativa es capturar el conocimiento de sentido común, los modelos cualitativos deben pasar, necesariamente, por representar de forma discreta -simbólica- los dos aspectos anteriores:

1) Caracterización abstracta de la noción de estado.
2) Formulación cualitativa de las leyes del comportamiento.

El primer aspecto persigue la representación de propiedades continuas del mundo mediante escalas discretas de símbolos. Esta forma de representación debe seguir siendo completa, de manera que cada posible estado relevante esté representado. Generalmente, un estado cualitativo se corresponderá con muchos estados en una descripción

clásica, caracterizados todos ellos por ser equivalentes en cuanto a su relevancia para el tipo de razonamiento que se realiza [Forbus 89].

Como consecuencia de la simplificación anterior, la capacidad predictiva de los modelos cualitativos es necesariamente menor, de forma que una interpretación de las leyes del comportamiento en terminos cualitativos tiende a ser generalmente ambigua e incluir respuestas espúreas. Sin embargo, a menudo, este tipo de respuesta puede ser suficiente. Además, aún cuando la respuesta ambigua no baste, el hecho de que la ambigüedad quede explícitamente representada tiene como ventaja el servir de guía apropiada para introducir nuevo conocimiento allí donde se necesita.

Un aspecto importante de la descripción cualitativa de la noción de estado es que permite, en ocasiones, la generación explícita de todos los posibles comportamientos de un sistema ("envisionment") de forma previa a cualquier razonamiento. Si bien la ambigüedad en la representación conduce a un grafo de estados, no hay un único estado siguiente a uno dado como ocurre en una simulación numérica, a cambio la inspección directa del grafo permite responder con poco trabajo a preguntas simples (por ejemplo, si se pretende saber si un determinado estado es alcanzable o no) cumpliéndose la máxima de DeKleer de que "un resolvedor inteligente de problemas debe ser capaz de responder a preguntas estúpidas y, preferiblemente, con menos trabajo del que llevaría responder a preguntas sutiles".

En lo que sigue se resumen los principales aspectos de la modelización del comportamiento, en concreto:

1) Diferentes formas de representar cantidades.
2) Representación cualitativa de ecuaciones diferenciales.
3) Paradigmas de modelización y razonamiento (ontologías).

La descripción se realiza en referencia, únicamente, a la Dinámica Cualitativa. Las vertientes Cinemática y Estática quedan fuera del ámbito de la tesis.

3.1.2. Representación cualitativa de cantidades.

En física clásica, los parámetros de estado de un objeto físico se representan, generalmente, como funciones reales de variable real. Una representación simbólica necesariamente debe discretizar la representación de números reales. Esto se hace principalmente de tres formas:

1) Mediante el signo de las cantidades.
2) Mediante desigualdades respecto de valores significativos para el comportamiento.

3) Mediante órdenes de magnitud.

El signo de una cantidad constituye su forma cualitativa más simple. Por ejemplo, el nivel del agua en un cauce de río podría tomar valores en la escala {-1,0,1} dependiendo de si el nivel es inferior, en el entorno de o superior a un umbral de normalidad. En algunos casos, esta simple asignación puede ser suficiente para capturar comportamientos relevantes.

A menudo, sin embargo, el interés está centrado en las direcciones de cambio de las cantidades, que vienen dadas por el signo de su derivada respecto del tiempo. DeKleer utiliza esta forma tan simple de representación para realizar análisis de perturbaciones en un sistema. El mayor problema de la representación mediante signo es que este se establece respecto de un único valor de referencia, siendo esta una aproximación poco realista.

La forma de comparar una cantidad con otros (varios) valores de referencia es mediante el concepto de espacio cuántico de [Forbus 81], como un conjunto ordenado (total o parcialmente) de desigualdades definidas en base a condiciones de contorno específicas de un determinado dominio. Nótese como el uso de desigualdades permite acumular de forma incremental información acerca de una cantidad. Supóngase, por ejemplo, el proceso de ebullición del agua y el comportamiento de su temperatura. De forma intuitiva, puede afirmarse que:

$$Tf < Ta$$
$$Ta > Tr$$
$$Ta < Te$$

siendo:

- Tf: Temperatura de fusión.
- Ta: Temperatura del agua.
- Tr: Temperatura del recipiente.
- Te: Temperatura de ebullición.

Las desigualdades anteriores definen un espacio ordenado parcialmente, a falta de saber la relación existente entre Te y Tr.

Si se aporta el conocimiento suficiente para establecer un orden total, la representación se simplifica, ya que se reduce a realizar una partición de $\Re$ en un conjunto de intervalos. Este es el caso de [Kuipers 86]. Bajo esta aproximación, se parte de la caracterización clásica de un sistema físico como un conjunto de parámetros reales variando continuamente en el tiempo. Cada parámetro es una función $f:[a,b]\rightarrow\Re^*$ que se comporta de forma "razonable", entendiendo esto como el cumplimiento de las siguientes condiciones:

1) f es continua sobre [a,b].
2) f es continuamente diferenciable sobre (a,b).
3) f tiene un número finito de puntos críticos en (a,b).
4) Lim f'(x) y Lim f'(x) existen y coinciden con f'(a) y f'(b) respectivamente.
   $x \to a$ $x \to b$

Las dos últimas condiciones excluyen funciones que oscilen infinitamente al acercarse a un punto dado interior al intervalo (condición tercera) o a sus extremos (condición cuarta).

El espacio cuántico, en este caso, se define como un conjunto finito de valores hito susceptibles de ser alcanzados por un parámetro, incluyendo siempre el cero, f(a), f(b), f(x) para cada punto crítico de f más, posiblemente, algún otro (la aproximación de Kuipers permite descubrir nuevos valores hito -no preestablecidos- en el curso de una simulación).

La representación de cantidades mediante desigualdades simples no siempre será suficiente. En ocasiones, no basta saber que X es mayor que Y, siendo necesario tener una estimación de en qué medida lo es. Dicha información puede actuar como una fuente de simplificación de un modelo, sin atentar de forma significativa contra los resultados de su ejecución.

Una forma de representar este tipo de conocimiento es potenciar el conjunto de relaciones posibles entre magnitudes [Raiman 86]; es decir:

$X << Y$: X es despreciable comparado con Y.
$X \sim= Y$: X y Y tienen valores muy próximos.
$X \sim Y$: X y Y son del mismo orden de magnitud.

El efecto más útil de estas relaciones es que producen una clasificación de valores en clases de equivalencia, por tanto una semántica precisa para proposiciones tales como "X es muy diferente a Y", "X es comparable a Y", etc. El principal problema es que este lenguaje para relacionar cantidades se obtiene después de establecer una correspondencia entre valores numéricos y ordenes de magnitud. Una forma de hacer esto es en términos de ratios. Por ejemplo, la relación X<<Y se puede definir como cierta cuando se verifica que:

$$|X|/|Y| < (1+c)$$

siendo c un parámetro específico del dominio.

3.1.3. Representación cualitativa de ecuaciones.

En física cualitativa, la representación de ecuaciones se realiza, de manera análoga a la representación de cantidades, por relajación de las ecuaciones clásicas. Este proceso de relajación da lugar a un conjunto de restricciones de tipo cualitativo que, de nuevo, son menos precisas, pero capaces de capturar conocimiento de tipo parcial, por un lado, así como de simplificar el proceso de inferencia que conduce a la predicción del comportamiento.

La estructura de un sistema se describe de esta forma mediante un conjunto de restricciones cualitativas a ser aplicadas al conjunto de parámetros representativos de su estado, de manera que actúan imponiendo límites a las posibles combinaciones de estados futuros. A menudo, dichas restricciones se expresan como predicados, en lugar de funciones, ya que su objetivo es examinar la consistencia de posibles asignaciones de valores a parámetros de estado.

Si el objetivo es representar ecuaciones ordinarias mediante restricciones cualitativas será necesario definir, al menos:

1) Restricciones aritméticas.
2) Restricciones funcionales.

Restricciones aritméticas son las correspondientes a los operadores básicos presentes en una ecuación: suma, resta, multiplicación y derivada. Por ejemplo [Kuipers 86] define de forma cualitativa los operadores aritméticos y diferencial mediante los siguientes predicados:

1) Dadas tres funciones "razonables" (según la definición del apartado 3.1.2) f,g,h: $[a,b]\rightarrow\Re^*$, SUMA(f,g,h) se verifica si, y solo si, $f(x)+g(x) = h(x)\ \forall x$ en [a,b].
2) Dadas tres funciones "razonables" f,g,h: $[a,b]\rightarrow\Re^*$, MULT(f,g,h) se verifica si, y solo si, $f(x) \cdot g(x) = h(x)\ \forall x$ en [a,b].
3) Dadas las funciones "razonables" f,g: $[a,b]\rightarrow\Re^*$, MENOS(f,g) se verifica si, y solo si, $f(x) = -g(x)\ \forall x$ en [a,b].
4) Dadas las funciones "razonables" f,g: $[a,b]\rightarrow\Re^*$, DERIV(f,g) se verifica si, y solo si, $f'(x) = g(x)\ \forall x$ en [a,b].

Aplicando las funciones anteriores es posible reducir una ecuación diferencial ordinaria a un conjunto de restricciones cualitativas. Por ejemplo, considérese la ecuación diferencial [Kuipers 86]:

$$x''(t) - x'(t) + \arctan(kx) = 0 \qquad (1)$$

equivalente a:

$$f_1 - f_2 + f_3 = 0 \qquad (2)$$

siendo:

$$\begin{aligned} f_1 &= f_2'(t) \\ f_2 &= x'(t) \\ f_3 &= \arctan(f_4) \\ f_4 &= kx \end{aligned} \qquad (3)$$

pueden obtenerse, finalmente, las restricciones cualitativas siguientes:

$$\begin{aligned} &\text{DERIV}(f_2,f_1) \\ &\text{DERIV}(x,f_2) \\ &\text{MULT}(k,x,f_4) \\ &\text{ADD}(f_1,f_3,f_2) \end{aligned} \qquad (4)$$

Los predicados (4) son matemáticamente equivalentes al conjunto de ecuaciones (2) y (3), excepto $f_3 = \arctan(f_4)$ que introduce la necesidad de representar dependencias funcionales mediante restricciones. A este fin, se introducen dos predicados adicionales, $M^+(f,g)$ y $M^-(f,g)$, de la forma siguiente:

- Dadas las funciones razonables f,g: $[a,b] \rightarrow \Re^*$, $M^+(f,g)$ se verifica si, y solo si, $f(x) = H(g(x))$ $\forall x$ en $[a,b]$, donde H es una función con dominio $g([a,b])$ y rango $f([a,b])$, diferenciable y cumpliendo que $H'(x) > 0$ en el interior del dominio. M^- se define de forma análoga pero cumpliendo que $H'(x) < 0$.

La definición anterior significa que $M^+(f,g)$ se verifica si f y g mantienen una relación monótona estrictamente creciente (decreciente para M^-). Esto es así ya que si $f(x) = H(g(x))$ entonces $f'(x) = H'(g(x)) \cdot g'(x)$ y, al ser $H'(x) > 0$ en el interior del dominio, f'(x) y g'(x) coincidirán en el signo de su derivada y en el cero.

De esta forma $f_3 = \text{arcatan}(f_4)$, en el ejemplo anterior, se reduce a la restricción cualitativa $M^+(f_3,f_4)$ que es menos restrictiva. De ello se deduce que toda solución de la ecuación (1) debe satisfacer también el conjunto de restricciones cualitativas pero no a la inversa.

Aunque todos los investigadores en física cualitativa definen tipos de restricciones para representar dependencias funcionales, su definición varía de unos casos a otros. Por ejemplo [Forbus 84] admite la definición de dependencias funcionales parciales mediante el concepto de "proporcionalidad cualitativa", con la notación:

$$Y \, \alpha_{Q+} \, X$$

que significa que Y = f(...,X,...) donde f es alguna función que crece monótonamente en su dependencia sobre X, asumiendo hipótesis de mundo cerrado (es decir, que la dependencia de Y sobre X es monótona creciente asumiendo que toda otra variable ha permanecido invariante). Dicha hipótesis es necesaria porque en la teoría de procesos cualitativos de Forbus las restricciones se asocian con procesos y únicamente cuando se conoce por completo el cuadro de procesos activos se sabe qué restricciones actúan y es posible calcular las dependencias funcionales. [DeKleer, Brown 84] expresan el mismo tipo de relación mediante una confluencia del tipo:

$$\delta x = \delta y$$

que se interpreta como:

$$Signo(x'(t)) = Signo(y'(t))$$

es decir, como una ecuación que impone igualdad en el signo de las derivadas. Dicha ecuación no implica, a diferencia de los predicados M y las proporcionalidades α_Q, la existencia de una relación funcional, siendo suficiente que el signo de las derivadas sea idéntico en todo t.

3.2. PARADIGMAS DE MODELIZACION Y RAZONAMIENTO.

Los paradigmas de modelización y razonamiento acerca del mundo físico tienen por objeto alcanzar, sobre este, abstracciones útiles y reusables; es decir, huyendo de la generación de modelos ad hoc para cada caso particular.

Como ya se comentó al principio del capítulo, se han propuesto diferentes tipos de abstracciones, entre las que cabe destacar:

1) El paradigma basado en restricciones de Kuipers ("Qualitative modeling").
2) El paradigma basado en componentes de DeKleer y Brown ("Envisioning").
3) El paradigma basado en procesos de Forbus ("QP Theory").

3.2.1. El paradigma basado en restricciones.

El enfoque de representación centrado en restricciones [Kuipers 85, 86] consiste, básicamente, en interpretar de forma cualitativa los modelos utilizados clásicamente para interpretación numérica; es decir, se basa en el tratamiento cualitativo de las ecuaciones diferenciales que se usan en la física tradicional para describir el comportamiento de un objeto particular. Así, de las ecuaciones que describen un cierto mecanismo se genera un conjunto de restricciones aritméticas y funcionales que representan estas ecuaciones y, a partir de las cuales, mediante un proceso de interpretación cualitativa, se crea un grafo de posibles transiciones entre estados cuyos caminos describen los posibles comportamientos del dispositivo a lo largo del tiempo.

El conocimiento para describir el comportamiento de un sistema físico es de dos tipos:

1) Conocimiento independiente del dominio. Es un conocimiento general acerca de como se concibe la evolución en el tiempo de cualquier sistema físico. La evolución de un sistema se define en base a las transiciones posibles entre estados cualitativos, dando lugar a un árbol de comportamientos a partir de un estado inicial.
2) Conocimiento específico. Conocimiento que describe la estructura física del sistema, entendiendo sobre las peculiaridades del comportamiento del mismo y sirviendo, por tanto, como mecanismo de poda del árbol de comportamientos. El conjunto de estados siguientes a uno dado se restringe a aquellos que sean consistentes con dicha estructura. Este conocimiento se especifica normalmente mediante restricciones cualitativas que constituyen versiones relajadas de las ecuaciones numéricas y diferenciales de la física clásica. Al igual que en ésta, los elementos ligados por las ecuaciones son características del sistema físico, definidas como funciones reales cuyo valor varía constantemente el el tiempo. Se impone que dicha variación debe producirse de forma "razonable" (ver apartado 3.1.3). Estas características son típicas de los sistemas reales.

La estrategia de control tiene por objeto deducir los posibles comportamientos futuros del sistema. Para ello se establece un proceso de simulación cualitativa a partir de restricciones que ligan las funciones descriptivas del sistema. A partir de un estado inicial, se produce un bucle de generación-filtrado de la siguiente forma:

1) Generación de nuevos posibles estados, de acuerdo con el conocimiento general de evolución de un sistema físico.
2) Filtrado de estados por el conocimiento dependiente del dominio, mediante aplicación de un proceso de satisfacción de restricciones tipo Waltz [Winston 84].

Como resultado se obtiene un grafo de transiciones de estado representando los posibles comportamientos futuros del objeto físico. Este grafo tiene un etiquetado temporal, de manera que en cada punto de tiempo se tiene un estado, o varios, relevantes para el comportamiento y definidos por una asignación de valor cualitativo a cada parámetro del sistema en ese punto de tiempo.

Esta forma de "envisionment" es capaz de detectar momentos, no previstos de antemano, en los cuales se producen cambios significativos. Por lo demás, el proceso de simulación se produce de forma análoga a la simulación numérica, excepto que los valores propagados vía la descripción estructural (ecuaciones) pertenecen a escalas cualitativas y están asociados con su dinámica de cambio (creciente, estable, decreciente). Finalmente, los puntos de tiempo considerados son solo aquellos que corresponden a estados del sistema cualitativamente distintos.

En resumen, el paradigma de Kuipers proporciona una forma de representar ecuaciones diferenciales ordinarias cualitativas, así como el método para hallar su solución o soluciones. Desde este punto de vista, debe considerarse una matemática cualitativa antes que una física cualitativa ya que el modelo de una situación física no está en correspondencia con la estructura real del objeto modelizado que, de esta forma, puede no ser reconocible fácilmente en el modelo.

3.2.2. El paradigma basado en componentes.

El enfoque centrado en componentes [DeKleer, Brown 84a, 84b], construye la representación apoyandose en la estructura del dispositivo objeto de modelización y de sus componentes.

La idea básica consiste en ver a un sistema como una colección de objetos (componentes). El comportamiento de un objeto se especifica según leyes internas, a menudo descompuestas en grupos de leyes según distintos estados o regiones de comportamiento del componente. Esta idea de región de comportamiento es análoga a la de región operativa de [Kuipers 86], si bien Kuipers la aplica a la totalidad del sistema, y no componente a componente del mismo.

Cada objeto tiene un cierto número de puertos de comunicación (conductos), de manera que toda interacción entre objetos transcurre a través de ellos. Dichos conductos definen, así mismo, la topología del sistema, dando lugar a una red de objetos.

La estructura física de un objeto, por tanto, se representa en un grafo en el cual los nodos son los **componentes** y los arcos **conductos** con transporte de un cierto material. Por ejemplo, para un circuito eléctrico, componentes sería resistencias, condensadores, etc, en tanto que los conductos serían los cables que conectan los componentes, formando el circuito. Finalmente, el material transportado sería la carga eléctrica.

El comportamiento se describe en términos de atributos del material; así, para un componente dado, es posible modelizar su comportamiento mediante las leyes que relacionan los atributos del material a su paso por el componente. Los atributos del material cambian de valor en los componentes (nunca en los conductos) de forma que, durante la modelización, terminan asociandose con los componentes en lugar de con los materiales. Así, se habla en términos de "Intensidad de corriente a su paso por la resistencia" en lugar de "Intensidad de la corriente" o del "Nivel en el contenedor" en vez del "Nivel del agua", etc.

Nuevamente, como en el caso del enfoque centrado en restricciones, se produce una interpretación cualitativa de las leyes del comportamiento, de manera que los atributos pueden tomar valor dentro de una gama discreta. Por su parte, la dinámica se representa por el signo de la derivada, tomando valores en el espacio cuántico {-,0,+}.

Al igual que en el modelo de Kuipers se tienen en cuenta dos tipos de conocimiento:

1) Conocimiento general, encargado de exigir que las variables descriptivas del sistema evolucionen en el tiempo bajo la hipótesis de continuidad.
2) Conocimiento específico, que recoge las condiciones relativas a los componentes y sus conexiones. Dichas condiciones se expresan mediante el concepto de confluencia que, en general, consistirá en una interpretación cualitativa de una ecuación diferencial. En este caso la confluencia se construye como adaptación directa del modelo físico convencional, pero también puede concebirse a partir del conocimiento de sentido común que se tenga sobre el comportamiento del objeto. Un modelo de comportamiento se representará, por tanto, como un conjunto de confluencias.

   Cabe considerar el caso en el que un único conjunto de confluencias no represente todo el comportamiento; es decir, que el dispositivo exhiba diferentes estados de funcionamiento, en cuyo caso habrán de definirse regiones de validez para cada estado y, en consecuencia, grupos separados de confluencias.

El proceso de inferencia tiene por objeto, al igual que la técnica basada en restricciones, deducir los comportamientos futuros del objeto modelizado. Esto se hace generando un grafo de "envisioning" que refleja las transiciones factibles entre los posibles estados. El proceso podría resumirse como sigue:

1) Identificación de confluencias activas (según el estado del dispositivo).
2) Obtención, mediante un proceso de satisfacción de restricciones, de valores de atributos consistentes con el conjunto de confluencias obtenido en 1).
3) Generación de nuevos estados, a transitar desde el estado actual, según las posibles evoluciones de las variables que satisfacen el conjunto de confluencias.
4) Para cada nuevo estado generado, volver al paso 1).

Una característica importante de este modelo es que, una vez construida una red de componentes, la mayor parte del proceso de inferencia es local al modelo, así como el comportamiento de cada objeto se deduce de forma independiente. Esto simplifica el proceso de razonamiento, a la par que permite desarrollar implementaciones

eficientes. Por contra, esa misma característica impide representar adecuadamente conocimiento de correlación entre el comportamiento de diferentes objetos; es decir, conocimiento que limita las posibles transiciones de estado de un objeto en virtud de los estados alcanzados por otro. Este aspecto es el que ha dado lugar a la concepción de las Tareas de Generación de Escenarios en esta Tesis.

[Forbus 89] identifica dos ventajas claras del modelo de DeKleer:

1) Su modularidad. El carácter local de las conexiones entre componentes permite la construcción, a priori, de modelos arbitrariamente grandes así como la definición, a tales efectos, de librerías de modelos estandar de componentes. Es posible construir este tipo de librerías siempre que sus componentes observen el principio de "non-function in structure"; es decir, que no se haga ninguna suposición sobre su estructura interna lo cual, paradójicamente, puede dificultar una descripción modular.
2) Su carácter derivado de los modelos tradicionalmente utilizados en ingeniería, para los cuales existen descripciones estructurales y modelos cuantitativos ampliamente aceptados y contrastados. Dichos modelos constituyen un valioso punto de partida para la construcción de modelos cualitativos, simplificando la tarea del diseñador.

La crítica principal al modelo de DeKleer estriba, además de en la dificultad ya mencionada para representar conocimiento de control dependiente del dominio global, en su falta de adecuación en dominios en los que no sea sencilla una descripción estructural (es decir, una correspondencia 1:1 entre objetos reales y componentes del modelo) o para modelizar sistemas cuyo comportamiento produce cambios en su topología (algún objeto desaparece). Sería muy difícil, por ejemplo, concebir un modelo de componentes para un sistema de combustión de gases en el interior de una tobera de un cohete, o un proceso de evaporación, etc.

3.2.3. El paradigma basado en procesos.

El enfoque centrado en procesos [Forbus 81, 84, 88, 89] se basa en la idea de que los sistemas físicos cambian de estado porque existe algún agente que induce ese cambio; es decir, existen procesos que provocan cambios en sus objetos componentes. Forbus aporta un esquema que permite representar tanto los procesos como los objetos influidos por ellos dentro de un sistema físico.

Bajo este paradigma, todos los cambios que se producen en un sistema físico se entienden causados, directa o indirectamente, por procesos. Un modelo de este tipo incluye una descripción de las clases de objetos existentes, y las relaciones entre ellos, así como las clases de procesos que podrían tener lugar. Un aspecto importante es que

el diseñador del modelo no necesita especificar qué procesos actúan en cada posible situación, sino únicamente las condiciones bajo las cuales las distintas clases de procesos han de mostrarse activos, corriendo a cargo del proceso de razonamiento generar instancias de procesos activos para cada situación particular.

Los elementos a representar son:

1) Cantidades.
2) Objetos, y sus estados.
3) Procesos, y sus influencias.

A partir de estos elementos de representación, la teoría explica cómo deducir los cambios posibles en el sistema.

Las **cantidades** representan parámetros que caracterizan el estado de un objeto a lo largo del tiempo. Una cantidad se caracteriza por su valor instantáneo y su derivada respecto del tiempo. Ambos atributos de la cantidad llevan asociados magnitud y signo. Los valores numéricos no se manejan explícitamente, sino referidos a un espacio cuantitativo ordenado según hitos o valores significativos. De esta forma, los cambios en los parámetros se reflejan mediante cambios en sus relaciones con el espacio cuantitativo. Esto permite representar fácilmente relaciones entre cantidades distintas que se muevan en el mismo espacio.

El **estado** de los objetos se representa mediante el concepto de **perspectiva individual**, que viene a ser una escena representativa del comportamiento de una colección de objetos relacionados, cuando se verifican ciertas condiciones.

Una perspectiva individual se define mediante los siguientes elementos:

1) Los tipos de objetos involucrados.
2) Un conjunto de precondiciones lógicas que deciden la factibilidad de la perspectiva.
3) Un conjunto de precondiciones cuantitativas relativas a las cantidades que definen el estado de cada objeto.
4) Un conjunto de relaciones que la perspectiva impone sobre los parámetros característicos de cada objeto.

El concepto de perspectiva permite, pues, declarar, qué hipótesis se pueden asumir como ciertas cuando una colección de objetos se halla en un cierto estado.

El concepto de **proceso** se define de forma similar al de perspectiva; es decir, mediante un proceso se caracteriza el estado de un conjunto de objetos y las relaciones que estos verifican pero, además, cuando el proceso está en curso "influye" activamente provocando cambios en los parámetros que definen el estado de los objetos involucrados.

Un objeto (cada uno de sus parámetros) puede estar influido por uno o varios procesos, de manera que, para cada parámetro, ha de cumplirse:

$$\delta C/\delta t = \sum_i I(P_i, C)$$

siendo:

C: cantidad asociada al parámetro.
P_i: conjunto de procesos influyentes.
t: tiempo.

es decir, la derivada de la cantidad respecto del tiempo es la suma de las influencias que sobre ella operan.

La interpretación de este esquema de representación permite obtener, a partir de la definición de las perspectivas y los procesos involucrados en un sistema físico, una imagen de sus posibilidades de evolución. Así, a partir de un estado dado, el procedimiento de inferencia a aplicar sería el siguiente:

1) Identificación de perspectivas activas.
2) Identificación de procesos activos.
3) Extracción del conjunto de influencias sobre cada parámetro en el que incide, al menos, un proceso activo.
4) Resolución de influencias, obteniendo, para las derivadas de los parámetros, valores consistentes con el conjunto de influencias. Estos valores determinan las posibles evoluciones del sistema.

Una vez concluido el proceso, es preciso realizar un análisis de límites para establecer si las precondiciones cuantitativas de los procesos y perspectivas activos se siguen verificando o si, por el contrario, no es posible mantener la configuración actual de procesos y perspectivas. Según esto, un proceso termina cuando se rebasan los límites impuestos por las desigualdades definidas por sus precondiciones cuantitativas.

El paradigma de Forbus es, en cierta forma, dual al de DeKleer. La teoría del proceso cualitativo es centrada en procesos, en tanto que el "envisioning" de DeKleer es centrado en objetos. Como consecuencia, los tipos de información disponibles localmente en una teoría se encuentran distribuidos en la otra [Bonissone, Valavanis 85];

por ejemplo, en la teoría basada en procesos el comportamiento de una variable viene dado por la acción conjunta de los procesos que sobre ella influyen. En la teoría basada en componentes, cada confluencia de un componente restringe el valor de las variables que participan en la confluencia y el cuadro total de confluencias de un componentes es interno al mismo. Por tanto, el conjunto de confluencias que actúan sobre una variable es también fijo y local al componente en el que reside la variable (excepto las confluencias de continuidad).

Las principales ventajas y desventajas relativas de ambos paradigmas, a la hora de modelizar, provienen de esa dualidad:

1) La noción de proceso es más útil para aquellos dominios en los que ciertos objetos pueden aparecer o desaparecer. Esto no está permitido en el paradigma centrado en componentes, que exige la construcción de una topología fija y completamente formalizada de forma previa a cualquier razonamiento.
2) El grafo de "envision" es muy útil para realizar razonamiento causal, ya que el flujo de información en el modelo se puede interpretar como flujo causal en el mundo modelizado. Sin embargo, este paradigma no permite establecer los limites del proceso físico y referirlo a cantidades. El paradigma de Forbus sí permite ese análisis de límites que, de hecho, es lo que da lugar a un grafo de transición de estados.
3) El paradigma basado en componentes tiene una importante ventaja: su descripción explícita de la estructura física del sistema. Esto facilita enormemente la tarea del constructor que, como ya se ha mencionado, además puede partir generalmente de modelos numéricos ya contrastados en la práctica de la ingeniería. No existe equivalencia, en este sentido, para el paradigma basado en procesos; es decir, no existe tradición modelística en ingeniería usando modelos orientados a procesos. De esta forma, al usar este paradigma el diseñador se ve obligado a identificar las clases de procesos (procesos abstractos) que actúan en una determinada escena física, y esa tarea puede no ser trivial.
4) El enfoque de Forbus es más flexible para modelizar conocimiento de tipo global (vía relaciones y precondiciones) para el cual el enfoque de DeKleer es más rígido. Es decir, en este último caso, el comportamiento del sistema completo se obtiene solo de forma parcial, ya que viene determinado por los comportamientos específicos de cada componente, no siendo posible hacer referencia, en los modelos locales de los componentes, a aspectos globales del comportamiento.

El paradigma centrado en restricciones no entra en la discusión anterior, ya que este enfoque no tiene en cuenta aspectos ontológicos en el sentido de [Forbus 89]. Es decir, el modelo requiere una completa lista de restricciones, referidas a todo el sistema, sin conexión con ninguna descripción de la estructura física real.

4. COMENTARIO GENERAL AL ESTADO DEL ARTE.

En resumen, los planteamientos de [Hayes 79, 85] y los avances registrados en el estudio de las técnicas de simulación cualitativa, debidos fundamentalmente a [DeKleer, Brown 84a, 84b], [Kuipers 84, 86] y [Forbus 84, 89], ponen de manifiesto que las técnicas de representación del conocimiento e inferencia puestas a punto para los tradicionales universos simples utilizados en la Inteligencia Artificial resultan insuficientes cuando se aplican a problemas relacionados con el mundo físico ya que, en este caso, no es sencillo formular ni adquirir unidades cognitivas acordes con las estructuras clásicas de representación (tipo reglas, operadores, jerarquías de marcos, etc). Por el contrario, en el mundo de la ingeniería profesional los problemas que se plantean y resuelven se basan en formas de entender y razonar sobre el comportamiento físico de objetos complejos sobre los que, en muchos casos, ya existen teorías clásicas para su modelización. Para el tratamiento de estos objetos la representación debe, por tanto, incorporar explícitamente el conocimiento existente, pero de forma implícita, en esos modelos, así como el conocimiento que, asimismo, los profesionales ponen en juego para la utilización de dichos modelos.

Las consideraciones anteriores conducen al diseño de representaciones con distintos niveles de detalle donde:

1) En niveles superficiales se manejen formas clásicas de representación y razonamiento.
2) En niveles más profundos se manejen representaciones y formas de razonar sobre comportamientos físicos.

Desde este punto de vista se alcanza, por tanto, un tipo de arquitectura que conjuga ambos niveles, incorporando formas de representación de la inteligencia clásica de resolución de problemas pero cuyas unidades cognitivas se pueden detallar a nivel profundo mediante representaciones de los fenómenos físicos relacionados con el tipo de problema a resolver.

Sobre una arquitectura de este tipo es posible producir entornos especializados por clases de problemas o campos profesionales, que se implementarían en base a colecciones de unidades cognitivas representativas de los distintos objetos físicos, de manera que el profesional solo tiene que aportar la linea general de razonamiento y los detalles (por especialización de las unidades cognitivas básicas) de su forma de entender y evaluar los fenómenos físicos involucrados en su caso de estudio.

Por otro lado, de la dificultad inherente a los esquemas clásicos de representación -a los que Chandrasekaran denomina lenguajes ensambladores para la representación del conocimiento- para estructurar la tarea de la representación nace el concepto de arquitectura modular, con razonamiento plurinivel, cuyo objetivo más

importante es definir formas de encapsular el conocimiento de manera que sea más abordable tanto su modelización como su uso y explicación.

Las primeras aproximaciones en el sentido de modularizar la representación se hallan en las arquitecturas de pizarra [Erman et al. 80] sofisticadas posteriormente en los aspectos de control por [Hayes-Roth 85], [Corkill et al. 88]. Como aproximaciones más recientes, y con mayor contenido a nivel cognitivo, se tienen la arquitectura para resolución universal de problemas propuesta por Newell en SOAR [Laird et al. 87] y las arquitecturas especializadas basadas en tareas genéricas debidas a [McDermott 89], [Chandrasekaran 87], [Brown, Chandrasekaran 89], [Chandrasekaran et al. 91] Estas últimas permiten abordar el problema de la representación en un dominio complejo mediante la división del método global de resolución en bloques genéricos que gozan de las características siguientes:

1) Desde el punto de vista funcional, constituyen abstracciones de cada parte del comportamiento global requerido.
2) Desde el punto de vista de la representación del conocimiento permiten utilizar formas genéricas de resolver partes del problema global ya identificadas previamente, así como la definición de nuevos bloques (tareas genéricas) para reutilización posterior, no solo desde el punto de vista de la instrumentación sino, además, como modelos para interpretación del conocimiento [Breuker, Wielinga 84a, 84b, 85], [Hicthman et al. 89].

Las tareas constituyen propuestas de organización del conocimiento que modelizan, a su vez, su forma de uso. Se entiende, por tanto, que un modelo del conocimiento no debe referirse únicamente al nivel del dominio sino, ademas, al nivel estratégico, siendo ambos niveles susceptibles de estructuración de forma diferente y característica según el tipo de problema a resolver. Esta mayor especificidad permite abordar la representación del conocimiento, heterogéneo, necesario cuando se tratan problemas del mundo real.

Como ya se señaló en el capitulo dos, el principal aspecto limitador de este tipo de arquitectura es el tratamiento simplista que en ella se hace del problema del control entre tareas, tanto en lo referente al secuenciamiento de tareas como a la comunicación de información entre las mismas, de suerte que el conocimiento de control no es objeto realmente de representación sino que se resuelve de forma prefijada. En problemas complejos relacionados con el mundo físico, esta concepción del control se revela claramente insuficiente en los dos aspectos reseñados. Efectivamente, asumiendo que la representación del comportamiento de los componentes involucrados en un sistema físico se resuelve mediante técnicas de razonamiento cualitativo:

1) No todos los componentes son relevantes en todo momento para explicar el comportamiento global y el criterio para focalizar la atención en los componentes importantes debe ser a su vez, por complejo, objeto de representación explicita.
2) Las influencias entre los componentes no se producen de forma determinista, siendo necesario definir conocimiento para su filtrado (criterios de verosimilitud, correlación entre componentes, utilidad según objetivos, etc) que puede ser lo suficientemente complejo como para requerir componentes de conocimiento representados explícitamente y específicos para este fin.

5. PROPUESTA DE TESIS: ARQUITECTURA COGNITIVA PARA REPRESENTACION DE CONOCIMIENTO PROFESIONAL.

La presente memoria propone una extensión de los trabajos de investigación de Chandrasekaran y consiste en el diseño y construcción de una arquitectura cognitiva para representación de conocimiento profesional especializado en clases de dominios relacionados con el mundo físico. La arquitectura que se presenta potencia el concepto de arquitectura basada en tareas genéricas, propuesto por Chandrasekaran, en dos aspectos concretos:

1) Propone una forma de encapsular el razonamiento cualitativo sobre el comportamiento de objetos físicos mediante tareas genéricas denominadas Tareas Básicas de Simulación (TBS). Esta linea de trabajo ya fue, de hecho, propuesta por el propio Chandrasekaran [Chandrasekaran, Milne 85], [Chandrasekaran 87] y permite avanzar hacia la construcción de librerías de tareas genéricas en el sentido de [Chandrasekaran 83], [Breuker, Wielinga 85] pero especializadas por áreas profesionales, donde un conjunto de tareas genéricas permita definir un modelo cognitivo de la forma de razonar del profesional en ese área.
2) Los entornos resultantes permiten un grado importante de programación de modelos en forma cognitiva por usuarios no especialistas en Informática.
3) Propone una forma de resolver el problema del control, con mayor adaptatividad a las formas cognitivas, siendo objeto de representación explícita a dos niveles:
   - A nivel profundo, mediante la definición de una tarea genérica denominada Tarea de Generación de Escenarios (TGE) para el control entre tareas que comparten información, actuando como filtros inteligentes que limitan la explosión combinatoria.
   - A nivel global, mediante la definición de una Tarea General de Control apoyada en una base de conocimiento ad-hoc, con objeto de centrar el razonamiento a nivel profundo (decidiendo que TBSs entran en juego) así como establecer criterios de utilidad o pertinencia a ser utilizados por las TGEs, de manera que el proceso de razonamiento cualitativo sobre el comportamiento se produce realmente guiado por objetivos, evitando la proliferación de comportamientos típica de las técnicas clásicas de simulación cualitativa.

La tesis propone, además, la forma de realizar la ingeniería del conocimiento acorde con las ideas anteriores de manera que, en el nuevo escenario, el ingeniero del conocimiento no necesita estructurar la representación por vía metodológica sino apoyado en los principios de organización del conocimiento facilitados por la arquitectura.

Los dos aspectos anteriores, arquitectura y construcción de modelos, conforman un enfoque completo para abordar la resolución de problemas complejos en el mundo físico, estructurada por clases de dominios. La figura 5a

muestra el ciclo de construcción de un resolvedor de problemas especializado en una clase de dominios, así como su utilización en un caso concreto de estudio.

La concepción del resolvedor parte necesariamente de un análisis a nivel del conocimiento [Newel 82]; es decir, se concibe como un agente inteligente cuya ley de comportamiento es el principio de racionalidad: el agente toma decisiones de cara a satisfacer sus objetivos. Por tanto, a este nivel la descripción es funcional, definiendose los objetivos y la naturaleza del conocimiento que el agente debe poseer para satisfacerlos.

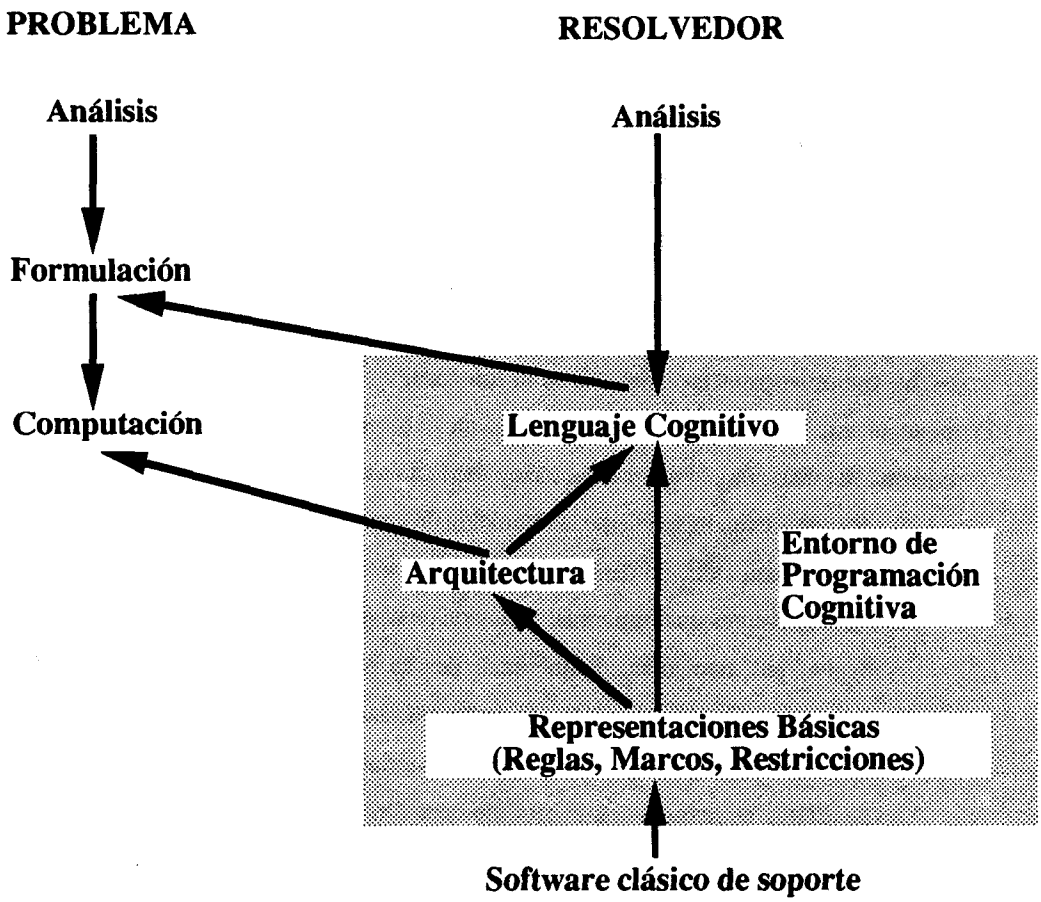


Figura 5a: Resolvedor de problemas en una clase de dominios.

La definición a nivel simbólico del agente especializado da lugar a un lenguaje cognitivo que, habiendo partido de la especificación funcional, debe ser necesariamente cercano a la forma de entender la resolución de problemas, en esa clase de dominio, desde el punto de vista más abstracto (o sea, sin alusión a las estructuras internas de representación del conocimiento ni a sus procedimientos de interpretación).

Por su parte, la implementación del lenguaje cognitivo requiere la creación de las estructuras de representación y procedimientos de interpretación necesarios, dando lugar a la arquitectura de representación del conocimiento, posiblemente apoyada en mecanismos básicos de representación.

Estos tres elementos -lenguaje, arquitectura y representaciones básicas- conforman, en cada clase de dominio, el entorno de programación cognitiva sobre el que apoyar la construcción de modelos. La definición de un entorno de estas características debe permitir al constructor de un modelo su formulación con gran analogía a la forma en que él entiende la resolución del problema, quedando oculta la representación interna del modelo así como su computación, ambos aspectos "comprendidos" por la arquitectura de representación.

5.1. SINTESIS DE LA ARQUITECTURA A NIVEL DEL CONOCIMIENTO.

Como ya se apuntó en el capítulo uno, el proceso general de razonamiento orientado a la toma de decisiones, en sistemas denominados en la literatura DSS ("Decisión Support Systems"), se puede definir mediante un agente cuyas capacidades serían [Cuena 89b]:

1) Predicción de la evolución de las acciones externas al sistema en un horizonte cercano de tiempo.
2) Identificación de las posibles transiciones futuras de estado producidas por el conjunto de acciones externas, asumiendo que se mantiene la estrategia de control actual.
3) Identificación de problemas actuales y predecibles a partir del estado actual y los estados predecibles deducidos en 2).
4) Inferencia de posibles nuevas decisiones que cambien la estrategia de control actual con objeto de resolver o mitigar los problemas detectados.

El conocimiento necesario para los puntos 3) y 4) se puede representar mediante paradigmas clásicos de clasificación y planificación produciendose elementos de conocimiento específicos en cada caso. Los puntos 1) y 2) tratan de la modelización del comportamiento, aspecto clave en sistemas físicos complejos. Según la rama de la ingeniería profesional de que se trate, se deben considerar soluciones para la representación diferentes y específicas.

En los casos en que el comportamiento general se puede explicar por un flujo de acciones entre diferentes componentes a lo largo del tiempo, el razonamiento sobre comportamiento puede describirse por una jerarquía de tareas de razonamiento específico de dos tipos:

1) Tareas Básicas de Simulación (TBS), que representan la respuesta de los diferentes objetos físicos que integran el sistema de flujo. El conocimiento interno de una TBS es un modelo cualitativo que relaciona el escenario de entradas al componente con posibles salidas en el tiempo.
2) Tarea de Generación de Escenarios (TGE), que representan el conocimiento para generar los escenarios de entrada para un tipo dado de TBS a partir de los escenarios de salida generados por otras TBSs anteriores en el flujo. El conocimiento en una TGE debe reducir el espacio de comportamientos a investigar teniendo en cuenta dos aspectos:

   - La *factibilidad* de las combinaciones entre los flujos generados por las TBSs entrantes.
   - La *relevancia* de las combinaciones a seleccionar con respecto a patrones de comportamiento cuya existencia se investiga. Estos patrones de comportamiento representan características relevantes para la evaluación del estado actual y la estrategia de control (por ejemplo, en el caso del estudio de los problemas de avenidas en cuencas hidrográficas, los comportamientos relevantes son aquellos que pueden producir avenidas cuando hay una situación de normalidad, o bien contemplan la reducción de niveles en una situación de desbordamiento).

La figura 5b muestra un ejemplo de una estructura de flujo de seis componentes. Los Componentes C1, C2 y C3 son del mismo tipo y su comportamiento general se representa por las instancias $TBS1^{(1)}$, $TBS1^{(2)}$ y $TBS1^{(3)}$ de la Tarea Básica de Simulación del tipo TBS1. $TBS2^{(1)}$ es la instancia de la tarea de tipo TBS2 que representa el comportamiento del componente C4. Finalmente, La tarea $TBS3^{(1)}$ es instancia de la tarea de tipo TBS3 y representa el comportamiento del componente C5.

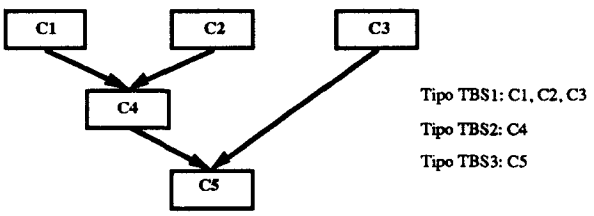


Figura 5b: Un sistema de flujo.

Según se desprende de la figura 5b, deben definirse tres TGE para control.

La figura 5c muestra parcialmente el árbol de razonamiento para generación de comportamientos a través de este sistema de flujos.

El árbol de exploración muestra el papel que juegan las TGEs para limitar su explosión combinatoria a base de producir solamente comportamientos factibles y relevantes para los objetivos. Sin embargo, el número de

comportamientos generados puede ser todavía excesivo ya que pueden existir muchos comportamientos relevantes con muy poca diferencia entre ellos. Esto justifica la necesidad de introducir un filtro adicional que asegure que los comportamientos propuestos sean también suficientemente diferentes para compensar el esfuerzo de su generación. Para cubrir este objetivo debe hacerse un retroceso inteligente, cada vez que se genera una alternativa de comportamiento, para seleccionar un nuevo escenario que produzca comportamientos, los cuales, siendo factibles y significativos de acuerdo con los objetivos establecidos, sean también diferentes en medida suficiente con respecto a los comportamientos ya generados. Este triple objetivo (relevancia, factibilidad y no redundancia) se puede satisfacer mediante un proceso de razonamiento superficial de tipo abductivo que genere, a partir de problemas potenciales, patrones generales de comportamiento que deben ser necesariamente respetados en fase de simulación.

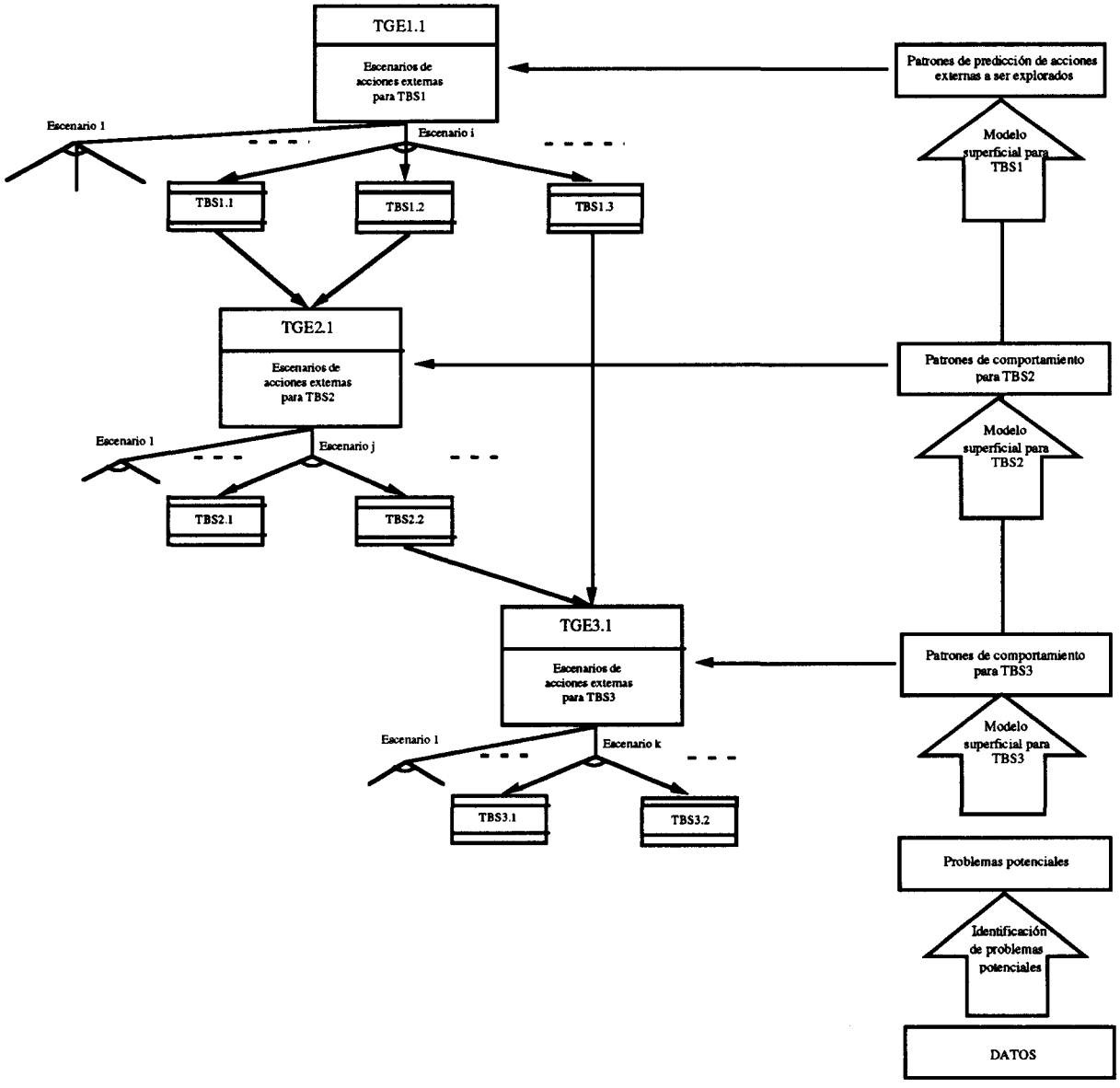


Figura 5c: Razonamiento sobre predicción de comportamientos.

Se trata, por tanto, de ensayar una forma de simulación guiada por objetivos en la que se rompe el esquema tradicional de simulación cualitativa, generadora del "envisionment" global de posibles comportamientos, con análisis posterior de su relevancia para la toma de decisiones. La simulación orientada a objetivos impone restricciones adicionales al proceso, de manera que se fuerza la producción de, únicamente, aquellos comportamientos de por sí interesantes desde un punto de vista estratégico.

En resumen, la solución propuesta se apoya en el criterio de simplificar, por un lado, la representación de las leyes físicas, lo que acarrea la aparición de un espacio de predicciones mayor, y por otro lado incluyendo criterios de factibilidad y relevancia para los objetivos, lo que debe reducir el espacio a explorar.

5.2. DESCRIPCION A NIVEL SIMBOLICO.

Acorde con las ideas anteriores, la arquitectura tiene la forma general propuesta en la figura 5d, donde se distinguen cinco tipos de tareas:

1) Tarea de Interpretación.
2) Tarea de Clasificación.
3) Tarea de Planificación.
4) Tarea General de Control.
5) Tarea de Simulación.

Los componentes de la arquitectura son, pues, módulos tarea cada uno de los cuales tiene una labor específica asignada y que se resuelven mediante paradigmas clásicos de representación y razonamiento, esquemas basados en tareas genéricas y, para la tarea de simulación, apoyado en técnicas de razonamiento cualitativo.

Cada componente de la arquitectura se describe por separado en los apartados siguientes.

5.2.1. Tarea de Interpretación.

A partir de una base de datos observados (por ejemplo, valores de nivel y caudal recogidos por sensores en una cuenca hidrográfica, valores de tensión y corriente en un circuito eléctrico, etc) la tarea tiene como objetivo realizar valoraciones cualitativas de esos datos dentro de escalas cualitativas prefijadas. Esta etapa de interpretación es necesaria como base para las etapas de razonamiento posterior, en particular para la Tarea de Simulación. La tarea se puede apoyar, a efectos de la representación del conocimiento, en las clásicas estructuras de marcos y reglas con estrategia de razonamiento guiada por objetivos o por datos, dependiendo del caso.

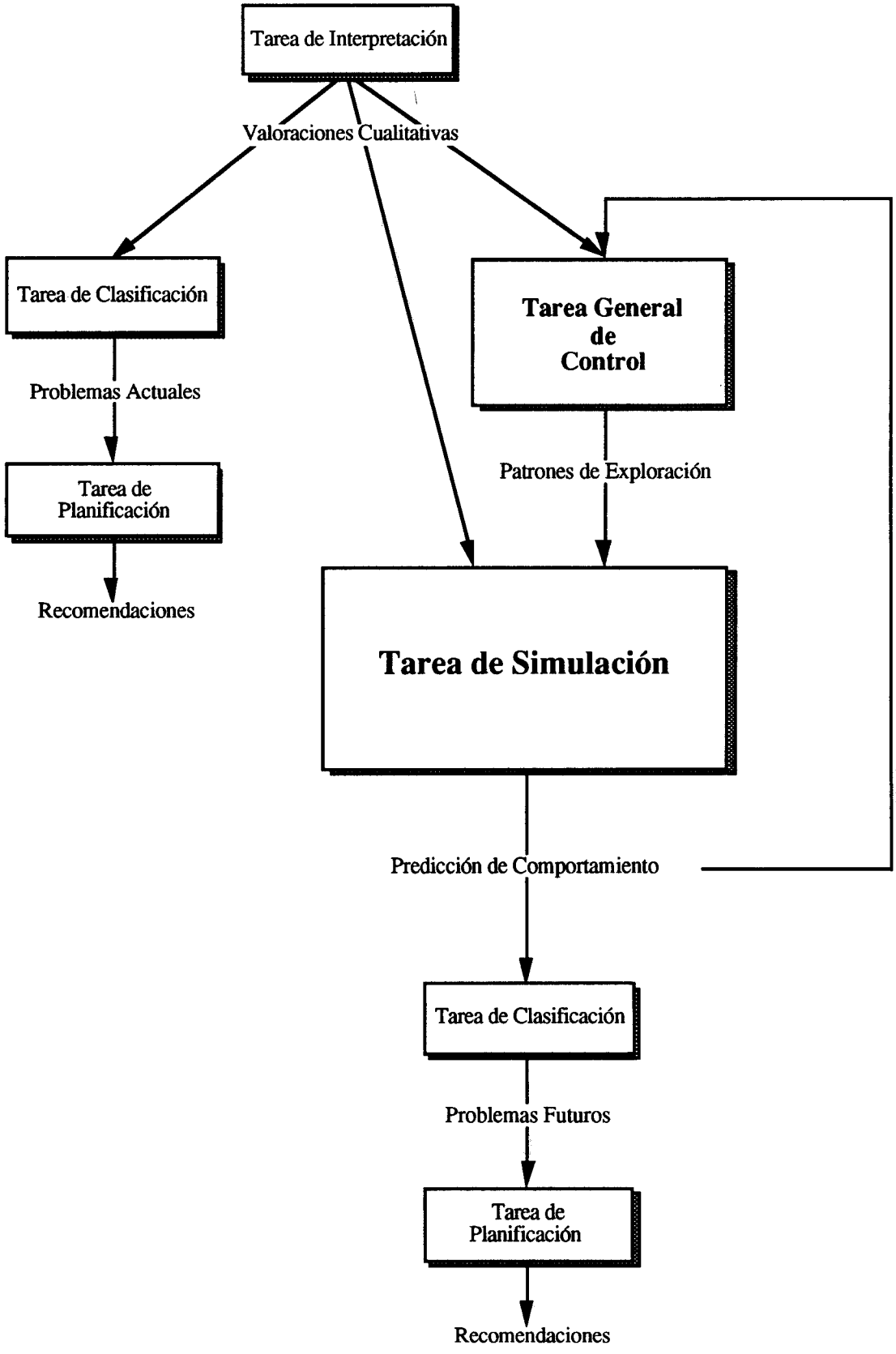


Figura 5d: Forma general de la arquitectura.

5.2.2. Tarea de Clasificación.

A partir de la descripción cualitativa del estado del sistema, la tarea tiene por objeto la identificación de los problemas a que ese estado puede dar lugar. La tarea opera en dos contextos diferentes:

1) Identificación de problemas actuales a partir del estado actual del sistema, dado como conjunto de valores cualitativos de variables obtenido como respuesta de la Tarea de Interpretación.
2) Identificación de posibles problemas futuros a partir de las transiciones de estado (escenarios alternativos de comportamiento futuro) generados por la Tarea de Simulación.

A efectos de la representación del conocimiento, la tarea puede basarse en un esquema de clasificación heurística [Clancey 85] o bien en un esquema de Tarea de Diagnóstico de tipo sistemático (vía relaciones causales, de descomposición, etc). La elección de uno u otro esquema dependerá de la calidad del conocimiento que se posea sobre el dominio. Cuando el conocimiento posea suficiente estructura como para que sea posible representar un modelo del mismo, la opción de diagnóstico sistemático será la más adecuada [Hitchman et al. 89]. En caso contrario la clasificación heurística constituirá la alternativa apropiada.

5.2.3. Tarea de Planificación.

A partir de los problemas identificados por la Tarea de Clasificación, la Tarea de Planificación tiene por objeto establecer planes de recomendación para eliminar los problemas presentados o, al menos, mitigar sus efectos (en el dominio de la predicción de inundaciones en las cuencas hidrográficas españolas, por ejemplo, los problemas de inundación por desbordamiento del cauce en las zonas bajas son inevitables, pues obedecen al ciclo hidrológico natural ante la presencia de una gota fría; sin embargo, sí es posible establecer con antelación suficiente los adecuados planes de control -cambios en la estrategia de control de embalses- así como las recomendaciones de protección civil, ambos dirigidos a mitigar los daños producidos por los problemas de inundación).

Como en el caso de la identificación de problemas, la tarea opera en dos contextos diferenciados:

1) Ante problemas actuales, derivados del estado actual del sistema.
2) Ante posibles problemas futuros, derivados de los escenarios de comportamiento futuro considerados factibles por la Tarea de Simulación.

La estructura de la tarea podría basarse en un esquema de Tarea Genérica de Diseño, si bien mostrando diferencias en cuanto a las entradas aceptadas por la tarea (conjunto de problemas, presentados como requerimientos de

entrada, en lugar de especificaciones funcionales) y las salidas producidas (secuencias de acciones en el tiempo en lugar de un modelo de diseño). En términos realistas, este tipo de tarea sería solo utilizable en el caso de existir un número alto de planes prefijados de actuación ante situaciones problemáticas concretas, siendo el objetivo de la tarea combinarlos adecuadamente. Para este caso, el esquema de Diseño Rutinario de [Brown, Chandrasekaran 89], presentado en el apartado 2.2.3.2, sería muy adecuado.

5.2.4. Tarea General de Control.

El objetivo general de esta tarea es supervisar el proceso de generación de escenarios futuros por parte del simulador, estableciendo límites a las posibles transiciones de estado que se pueden producir y acotando, por tanto, el pasillo de búsqueda. Esta labor de supervisión contempla tres aspectos diferenciados:

1) Focalizar la atención de la Tarea de Simulación en la dirección adecuada; es decir, identificar qué componentes del sistema van a contribuir de forma relevante al comportamiento global y, por tanto, su comportamiento particular debe ser estudiado por vía de razonamiento profundo.
2) Establecer criterios de filtrado a ser utilizados por el simulador cualitativo, de manera que éste pueda generar como comportamientos solo aquellos que, potencialmente, pueden derivar en problemas.
3) Establecer restricciones adicionales, en cada simulación, de forma que el simulador solo busque comportamientos significativamente diferentes de los ya generados anteriormente.

Cada uno de los aspectos anteriores incide en el objetivo común de evitar esfuerzos inútiles de simulación cualitativa (más costosa).

En conjunto, la Tarea General de Control supone una versión superficial (heurística) del propio simulador cualitativo ya que, a partir del estado en curso (ya sea el estado actual o los posibles estados futuros producidos por la Tarea de Simulación Cualitativa) la tarea entrega un conjunto de patrones de comportamiento que, en sí mismos, ya constituyen una primera previsión del comportamiento global del sistema. Si bien esta previsión no puede entenderse como respuesta definitiva, si puede actuar como base para identificar los criterios de poda mencionados y evitar que la Tarea de Simulación Cualitativa se dirija hacia la generación de comportamientos irrelevantes desde el punto de vista de la toma de decisiones.

Esta forma de entender el razonamiento profundo guiado por patrones ya se recoge en [Alonso et al. 90a,90b] y, posteriormente, en [Chandrasekaran et al. 91] y [Alonso et al. 92a].

Para poder generar los patrones de comportamiento final a explorar por la Tarea de Simulación Cualitativa, la Tarea General de Control debe contener un modelo simplificado del sistema y de como se distribuyen los

problemas espacial y temporalmente en el dominio de aplicación. A efectos de la representación del conocimiento la tarea puede basarse en la estructura de un sistema de mantenimiento de verdad de acuerdo con las propuestas de [Doyle 79, 82], en el cual las proposiciones básicas sean el grado de interés de distintas hipótesis de comportamiento relacionadas según una estructura de grafo dirigido pseudo-arboriforme. Una hipótesis interesante da lugar a que un conjunto determinado de patrones de comportamiento final sean introducidos en el simulador cualitativo. La actuación de la Tarea de Simulación verificará o no las hipótesis seleccionadas, lo cual servirá a la Tarea General de Control para revisar el grado de interés de las diferentes hipótesis.

En el capítulo 6 (apartado 6.1.1) se describe con detalle la implementación de esta tarea.

5.2.5. Tarea de Simulación Cualitativa.

Esta tarea tiene por objetivo establecer posibles evoluciones del sistema físico -caso de estudio- en distintos horizontes temporales. Bajo la hipótesis de un sistema de flujo, el horizonte de predicción puede ser deducido a partir de los horizontes temporales considerados para cada objeto componente. En esencia, pues, el objetivo es obtener una descripción del comportamiento del sistema físico en base a sus transiciones de estado, de forma totalmente acorde con las propuestas comentadas en el capítulo tres. Sin embargo, en este caso no se pretende obtener un grafo de "envisioning" total sino un árbol parcial con únicamente transiciones que, de acuerdo con los criterios de control mencionados, deriven en estados que sean fuente potencial de problemas. Para ello, la tarea acepta como entrada no solo la descripción del estado actual producida por la Tarea de Interpretación sino también los patrones de comportamiento final deducidos por la Tarea General de Control.

La arquitectura interna de la tarea se muestra en la figura 5e, que pretende reflejar la profundización del conocimiento sobre la forma de entender el comportamiento de los distintos elementos involucrados en el sistema de flujo.

La arquitectura se construye en base a las influencias en cascada de unos componentes sobre otros en distintos niveles y mediante, a su vez, dos tipos de tareas:

1) Tareas Básicas de Simulación (TBS). Encargadas de encapsular el razonamiento cualitativo sobre cada objeto físico individual. Como abstracción funcional, la TBS puede entenderse como una macro-regla deductiva con el formato siguiente:

$$Ax_i, E_i \Longrightarrow C_i$$

siendo Ax_i una predicción de acciones externas al objeto a lo largo del tiempo, E_i un estado inicial y C_i una predicción de evolución de estados en el tiempo como respuesta al conjunto de solicitaciones. En el caso general, C_i será un árbol parcial de "envisionment" con el estado E_i como raíz y que puede describirse por vértices con etiquetado temporal o bien mediante valores hito en el sentido de [Kuipers 86]. Aunque el razonamiento dependiente del tiempo es un aspecto que todavía requiere investigación y se sale del ámbito de esta tesis, en sistemas de flujo sin retroalimentación es posible realizar estimaciones temporales en base a características concretas de los objetos simulados, características que permiten asignar al menos intervalos de ocurrencia de los estados a lo largo de cada camino del árbol de estados generado [Cuena 88a].

Dependiendo de la naturaleza del objeto simulado, y del nivel de teoría que sobre él se tenga, la representación del conocimiento puede resolverse mediante un modelo superficial que refleje explícitamente las posibles transiciones de estado del objeto, o bien mediante un modelo cualitativo en el sentido de [Kuipers 86], [DeKleer, Brown 84a, 84b] o [Forbus 84]. En los casos en que se parta de un modelo profundo del segundo tipo, cabe plantearse generar muestras representativas del comportamiento del objeto y aplicar a dichas muestras técnicas de aprendizaje automático del tipo [Quinlan 79] o [Michalski et al. 86] para pasar de un nivel a otro de modelización, en linea con las propuestas de [Steels 89].

En cualquier caso, y con independencia de la forma de interpretación a nivel básico, la especificación funcional será, en general, la misma para los componentes del sistema que sean del mismo tipo. Para el usuario constructor de un modelo, al definir un componente concreto, solo será necesario en estos casos adscribir el componente al tipo que corresponda.
En el capítulo 6 (apartado 6.2) se describe en detalle la implementación de esta tarea.

2) Tareas de Generación de Escenarios (TGE). Encargadas de expresar la forma de acumular acciones de unos componentes sobre otros. A partir de los comportamientos individuales generados por un conjunto de componentes, la tarea produce un primer escenario de solicitación (asignación individual de valores a variables) para actuar como acciones externas sobre los componentes por ellas influenciados al siguiente nivel de simulación. La tarea albergará conocimiento de las siguientes clases:
   - Conocimiento de la topología y cronología del sistema objeto de modelización; es decir, de la forma de conexión de los componentes en términos de las variables que comparten. El formato general de una relación representada mediante una TGE sería el siguiente:

$$AP_i, C_j, C_s, ..., C_p \Longrightarrow Ax_i$$

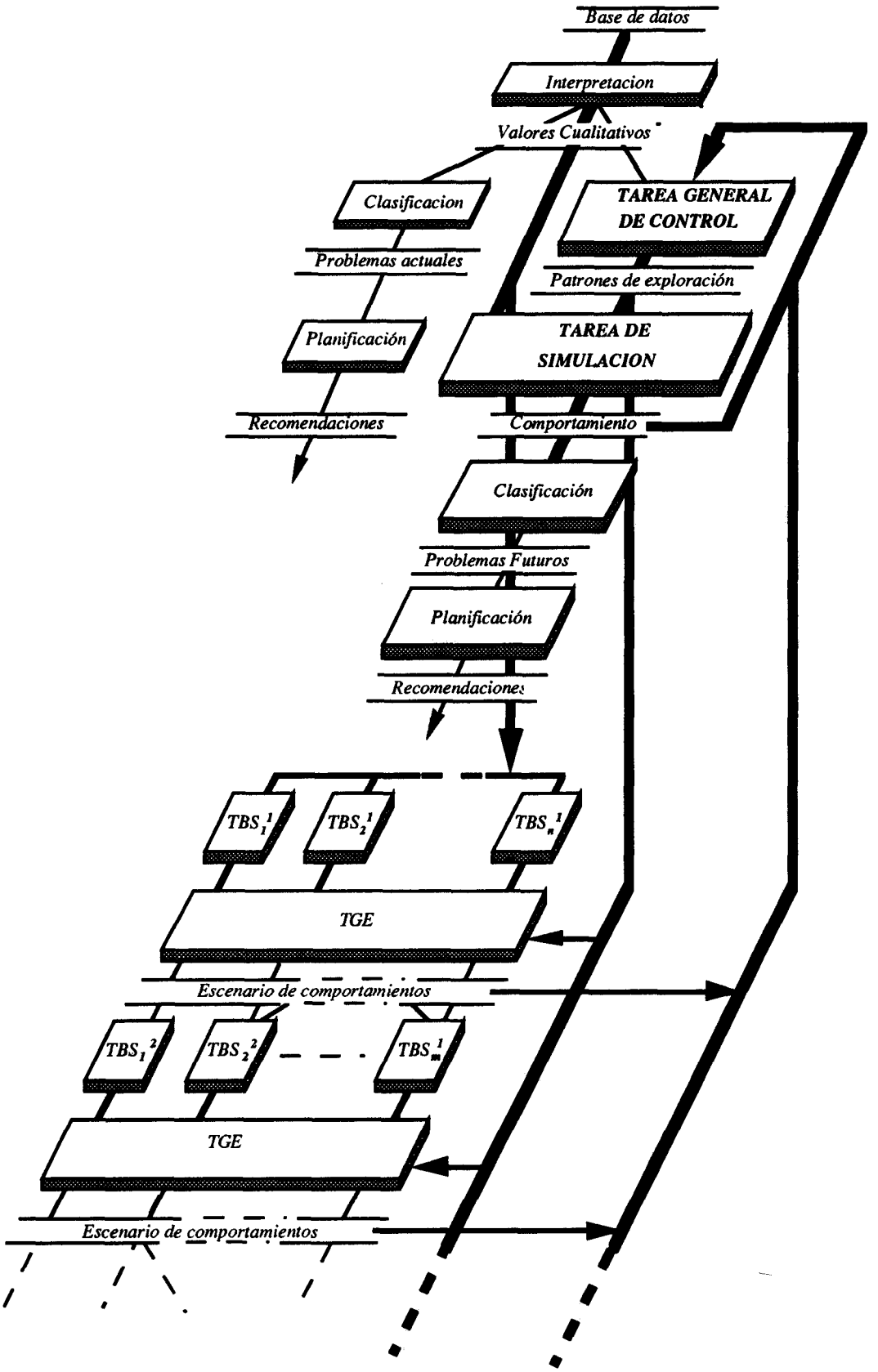


Figura 5e. Tarea de Simulación y Profundización del conocimiento.

siendo C_k comportamientos de otros componentes incidentes como acciones externas sobre el componente i, AP_i las acciones externas directas del medio sobre i y Ax_i el escenario de solicitación generado por la TGE para dicho componente.

- Conocimiento de correlación entre los componentes entrantes; es decir, criterios de verosimilitud aplicables a los comportamientos de entrada como mecanismo de filtrado, de manera que solo se generen escenarios considerados verosímiles.
- Conocimiento de pertinencia por objetivos; es decir, criterios de factibilidad según objetivos de búsqueda definidos por patrones de comportamiento final suministrados por la Tarea General de Control, de manera que solo se generen escenarios que entren dentro de los limites impuestos por esos patrones.

A efectos de la representación del conocimiento, la TGE se puede resolver mediante bases de reglas (para deducir umbrales de crecimiento y decrecimiento de variables, asignar valores a variables intermedias, etc) y mediante sistemas de satisfacción de restricciones (para la representación de los criterios de control arriba mencionados).

En el capítulo 6 (apartado 6.2.1) se describe en detalle la implementación de esta tarea.

A nivel superficial, el modelo de razonamiento contenido en la Tarea de Simulación Cualitativa puede interpretarse por un proceso de propagación de comportamientos a través de una red de objetos interrelacionados y sin ocurrencia de "feedback". La transmisión de comportamientos se produce a diferentes niveles, cada nivel definido por la actuación de una TGE.

El proceso de simulación se describe por un bucle con dos pasos (nivel i):

1) Ejecución en bucle de las Tareas Básicas de Simulación del nivel correspondiente. Cada TBS produce, como respuesta individual, una gama de comportamientos alternativos.
2) Ejecución de la TGE correspondiente al nivel de simulación. La TGE restringe los comportamientos entrantes a un único representante por cada TBS, conformando el escenario de solicitación (acciones externas) para las TBSs en el siguiente nivel. Para el primer nivel, el conjunto de acciones externas a las TBSs serán las acciones externas del medio sobre el el sistema global.

Los pasos 1) y 2) se repiten hasta haber recorrido toda la estructura, siempre y cuando ninguna TGE haya generado el escenario vacío, lo que se interpreta por infactibilidad de las combinaciones de comportamientos entrantes; es decir, los criterios de filtrado impuestos pueden hacer que alguna TGE no genere ningún escenario

(combinación) de solicitaciones para el siguiente nivel de simulación, ya sea por criterios de verosimilitud o pertinencia por objetivos o ambos. Esto equivale a decidir la infactibilidad del escenario de exploración propuesto por la Tarea General de Control.

La unión de los escenarios producidos en los diferentes niveles de simulación constituye la respuesta global de la Tarea de Simulación. Dicha respuesta es realimentada como entrada a la Tarea General de Control, para que ésta deduzca nuevos criterios de control.

5.2.6. Estrategia general.

La estrategia general de razonamiento a partir de estos conceptos se basa en inferir problemas potenciales que luego han de ser verificados en simulación. Para ello, la estrategia general se apoya en modelos profundos capaces de simular el comportamiento de cada objeto físico y en un modelo superficial que entiende de como rentabilizar los esfuerzos de simulación así como interpretar los resultados y tomar decisiones en consecuencia, emulando a un profesional especializado en la problemática planteada en el dominio de aplicación y que, en consecuencia, explota inteligentemente dichos modelos.

Se distinguen dos lineas separadas de razonamiento:

1) Linea de evaluación: identificación de posibles problemas derivados de la situación actual y planificación de acciones de control acordes con dichos problemas.
2) Linea de predicción: predicción, por simulación cualitativa, de posibles escenarios futuros de comportamiento y evaluación de la misma forma que para el estado actual.

La estrategia de razonamiento global se estructura en las siguientes etapas:

1) Tarea de Interpretación: identificación del estado actual por interpretación cualitativa de los datos observables de entrada.
2) Linea de evaluación.
    2.1) Tarea de Clasificación: identificación de posibles problemas a partir del estado actual.
    2.2) Tarea de Planificación: planificación de acciones de control para eliminar los problemas identificados en el paso anterior o, al menos, mitigar sus efectos.
3) Linea de predicción.
    3.1) Tarea General de Control: generación, desde el estado actual, de hipótesis de problemas futuros para actuar como patrones de comportamiento final a ser explorados (verificados) en simulación cualitativa.

3.2) Tarea de Simulación Cualitativa: generación de un escenario de comportamiento futuro acorde con los patrones introducidos por la Tarea General de Control, según la estrategia presentada en el apartado 4.2.5.

3.3) Evaluación del escenario generado de forma análoga a la ya realizada para el estado actual; es decir:

3.3.1) Tarea de Clasificación: identificación de posibles problemas futuros.

3.3.2) Tarea de Planificación: planificación de acciones de control.

El paso 3.3) no se realiza si la Tarea de Simulación Cualitativa no ha sido capaz de generar ningún escenario acorde con los patrones de comportamiento final.

3.4) Tarea General de Control: a partir del escenario de comportamiento obtenido en simulación (o del escenario vacío en su caso) y el conjunto actual de patrones de comportamiento final, se plantea la selección de una nueva hipótesis de exploración; es decir:

* Nuevo conjunto de objetos físicos cuyo comportamiento ha de estudiarse por simulación cualitativa.
* Nuevo conjunto de patrones de comportamiento final a explorar.
* Restricciones adicionales como entrada para las Tareas de Generación de Escenarios.

Con la nueva hipótesis obtenida se vuelve al paso 3.2).

El proceso continúa en bucle entre los pasos 3.2) al 3.4) hasta que la Tarea General de Control no genere ninguna nueva hipótesis de exploración o bien se alcance un límite de tiempo prefijado.

6. IMPLEMENTACION DE LA ARQUITECTURA.

6.1. SOLUCIONES AL PROBLEMA DEL CONTROL.

La arquitectura propuesta se estructura, desde el punto de vista estratégico, en distintas etapas de razonamiento encadenadas. Cada etapa, por tanto, es premisa para las siguientes. Respecto del control se plantean dos problemas diferenciados:

1) Los resultados de cada etapa de razonamiento no tienen por qué producirse de forma determinista, ya que es posible considerar como factibles diferentes opciones alternativas con diversos grados de certeza.
2) A nivel global, pueden producirse gamas de comportamientos cualitativamente equivalentes o bien irrelevantes desde el punto de vista de la toma de decisiones.

El primer aspecto exige una supervisión cualitativa de las conclusiones intermedias obtenidas durante el proceso de razonamiento, dando lugar a una forma de control localizado con dos objetivos principales:

1) Filtrado de resultados intermedios según criterios de utilidad o pertinencia. Dichos criterios pueden venir impuestos a nivel general y, utilizados localmente, dan lugar a que solo se tengan en cuenta aquellas alternativas de comportamiento de cada objeto que, potencialmente, puedan contribuir a comportamientos globales de interés para la toma de decisiones.
2) Filtrado de resultados intermedios según criterios de verosimilitud; es decir, que restrinjan el número de combinaciones eliminando aquellas que, por el conocimiento que se tiene del dominio, resulten poco creíbles.

El segundo aspecto exige capacidad para centrar el proceso de generación de comportamientos en la búsqueda de aquellos relevantes para la política de control, dando lugar a una forma de control global.

Dicha capacidad se traduce en acuñar criterios de pertinencia, a ser explotados después en las etapas intermedias de razonamiento, a partir de información sobre el estado actual del sistema. Paralelamente, este control global debe evaluar los comportamientos préviamente generados, de manera que los criterios de pertinencia incluyan también aspectos que primen la obtención de comportamientos significativamente diferentes de los ya obtenidos, evitando así esfuerzos inútiles en las etapas de razonamiento profundo.

6.1.1. La Tarea General de Control (TGC).

La Tarea General de Control (TGC) es el módulo cognitivo que controla a nivel global el espacio de búsqueda explorado por el sistema. Esto significa que la TGC tiene como misión imponer la estrategia global a seguir en simulación cualitativa. A nivel de detalle (en las Tareas de Generación de Escenarios) se toman decisiones de tipo táctico con objeto de cumplir la estrategia general.

Básicamente, la TGC constituye un sistema experto en identificar patrones de comportamiento que deberán ser tenidos en cuenta en simulación cualitativa para la generación de escenarios solución. Esta forma de exploración de comportamientos guiada por patrones ya ha sido propuesta por [Alonso et.al 90a, 90b], [Chandrasekaran et al. 91] y [Alonso et.al 92a].

El esquema de inferencia de la TGC es el siguiente:

$$P, C \Longrightarrow P'$$

siendo:

P: última conjunción de patrones utilizada durante el proceso de generación de escenarios solución.
C: comportamiento deducido por el modelo profundo (escenario que verifica P).
P': nueva conjunción de patrones como entrada de la próxima simulación.

Es decir, dada la conjunción de patrones utilizada en simulación y el comportamiento deducido, la TGC propone una nueva conjunción de patrones.

Para su operación, la TGC se apoya en una base de conocimiento sobre distribución espacial y temporal de patrones así como su forma de agregación formando hipótesis potencialmente interesantes para su exploración. Cada hipótesis define un área de interés compuesto por una determinada agrupación de objetos relacionados dentro del sistema de flujo. Para que este esquema sea aplicable debe ser posible descomponer el dominio en áreas (y subáreas dentro de las áreas, etc) en las que, eventualmente, pueda centrarse la simulación.

En las apartados siguientes se describe la representación del conocimiento y la estructura de la tarea.

6.1.1.1. Representación del conocimiento.

La base de conocimiento de TGC contiene conocimiento de como se distribuyen (potencialmente) los problemas en el sistema objeto de modelización. Esta distribución se hace a distintos niveles de detalle. El nivel inferior lo constituyen patrones elementales asociados a las variables básicas que describen el comportamiento del sistema. Los niveles intermedios están constituidos por hipótesis de patrones agregados compuestos, a su vez, por conjunciones de patrones elementales, conformando escenarios de posibles focos de problemas. En el apartado 6.1.1.3. se describe la estructura de los patrones de comportamiento final.

La forma de descomponer el sistema en hipótesis de estudio, subhipótesis y patrones elementales puede entenderse como una estructura pseudoarboriforme en la que los nodos terminales son patrones elementales y los nodos intermedios hipótesis más abstractas de patrones agregados (ver figura 6a). La característica pseudoarboriforme viene dada por el hecho de que un nodo cualquiera puede tener mas de un padre en la estructura.

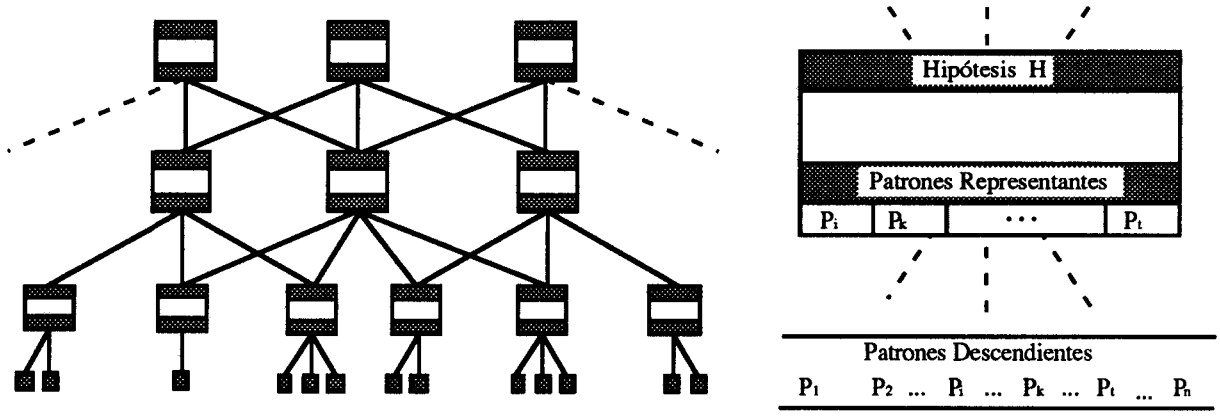


Figura 6a: Estructura del árbol de hipótesis de la TGC.

Por su parte, la abstracción se produce porque una hipótesis intermedia no tienen porqué albergar, en general, a la totalidad de los patrones elementales que son hojas descendientes de ella en el árbol, sino únicamente un subconjunto que actúa como representante de esa totalidad. Para ello, los patrones representantes, cuya agregación constituye una hipótesis de exploración, se eligen de forma que las variables controladas por esos patrones estén influenciadas en el tiempo por las variables asociadas a los patrones pertenecientes a hipótesis descendientes en el árbol, de manera que se pueda asumir que resumen su comportamiento. Este esquema puede ser particularmente aceptable en sistemas de flujo sin "feedback", donde las variables alineadas en el orden del flujo se influencian en un único sentido. Por ejemplo, en una red de transporte rápido de aguas en una Cuenca Hidrográfica, los valores de nivel y caudal en cada punto de la red propagan su influencia aguas abajo. Considerando su comportamiento en el tiempo, es factible plantear que los patrones que controlen dichas variables tengan como patrón representante aquel que tenga como variable asociada de nivel y/o caudal la que corresponde al punto de drenaje de la red.

Una hipótesis que se usa en simulación representa un patrón general de búsqueda de problemas potenciales en el área (colección de objetos del sistema) influida por el conjunto de sus patrones elementales descendientes en el árbol de hipótesis. Propuesta una hipótesis por la TGC, la Tarea de Simulación busca generar un escenario de comportamiento que la verifique (confirmación de la hipótesis). La actuación del simulador arrojará como resultado que la hipótesis se confirme total (todos los patrones descendientes se verifican) o parcialmente (algún patrón descendiente se verifica). En base a este objetivo, la estructura de la base de conocimiento es tal que se cumplen dos condiciones:

1) Si la TGC descarta una hipótesis para su uso en simulación, todas las hipótesis descendientes de ella en el árbol quedan igualmente descartadas.
2) Dada una hipótesis confirmada parcialmente, la estructura del árbol indica qué hipótesis descendientes deben considerarse para que los comportamientos a generar por el simulador sean significativamente diferentes de los ya generados anteriormente.

Naturalmente, la TGC puede encontrarse dos hipótesis compitiendo por entrar en el simulador. Para discriminar hipótesis candidatas se define una función de interés que tiene en cuenta dos aspectos:

1) Superficie abarcada por la hipótesis. Dependiendo del dominio, este atributo se interpretará de forma diferente, aunque su objetivo siempre será establecer una medida cualitativa del tamaño del foco de atención que ha de considerarse en simulación cualitativa; es decir, qué objetos deben estudiarse en fase de razonamiento profundo.
2) Gradiente de cambio impuesto por la hipótesis. Este gradiente es una medida de la distancia entre la Valoración Actual y la Valoración Prevista de la variable controlada por cada patrón de la hipótesis (ver apartado 6.1.1.3).

La función de interés de una hipótesis se puede definir como la suma de las funciones de interés de sus patrones representantes. No obstante, esta cuestión debe siempre quedar abierta, de manera que el constructor del modelo pueda establecer el criterio.

6.1.1.2. Estructura de la tarea.

El objetivo básico de la tarea es supervisar el proceso de generación de escenarios de comportamiento futuro en simulación cualitativa. La figura 6b muestra la estructura general de la tarea.

El motor de inferencia de la TGC consiste en un procedimiento de interpretación del árbol de hipótesis, con objeto de decidir qué hipótesis deben ser tenidas en cuenta por el simulador en cada momento.

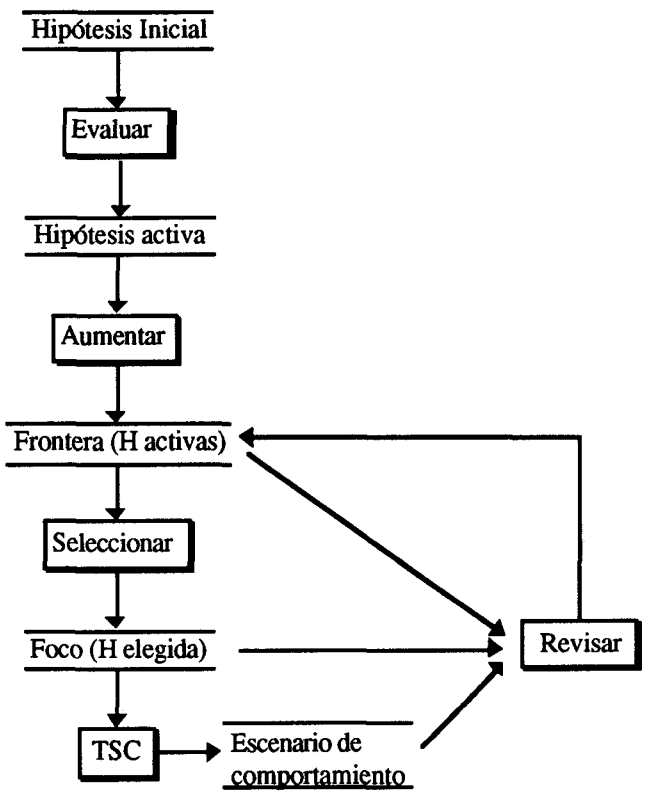


Figura 6b: Estructura de la Tarea General de Control.

El motor opera, básicamente, en tres fases:

1) Evaluación de hipótesis.
2) Selección de una hipótesis.
3) Revisión del árbol de hipótesis.

La evaluación de una hipótesis consiste en establecer si se considera relevante para su estudio en simulación. La relevancia se investiga por medio de un atributo denominado Nivel de Activación que toma valores en un dominio del tipo <*Activa, No activa, Posible*>. Estos valores se interpretan de la forma siguiente:

1) *Activa*: indica que la hipótesis es relevante y debe tenerse en cuenta en el momento de seleccionar la conjunción de patrones que actuará en el simulador.
2) *No activa*: indica que la hipótesis representada es irrelevante y se descarta para la selección.

3) *Posible*: indica que la hipótesis, en si misma, no es relevante pero alguna de sus subhipótesis puede serlo.

La forma de deducir el nivel de activación debe poder ser impuesta por el usuario constructor del modelo para cada hipótesis. El criterio por defecto puede establecerse de acuerdo con las siguientes reglas:

Si *todos los patrones representantes son activos*
Entonces l*a hipótesis es Activa*.

Si *algún patrón representante es activo*
Entonces l*a hipótesis es posible*.

Si *ningún patrón representante es activo*
Entonces *la hipótesis es No activa.*

La selección de una hipótesis consiste en elegir, entre las hipótesis activas obtenidas durante la evaluación, aquella que será entrada del simulador. Para discriminar qué hipótesis resulta elegida se utiliza la función de interés ya descrita.

Elegida una hipótesis, el simulador buscará confirmarla apoyandose en las restricciones impuestas por los patrones representantes de la misma. Así mismo, confirmar una hipótesis consiste en verificar los patrones descendientes de ella en el árbol de hipótesis.

Las restricciones que actúan en las TGEs están definidas realmente para todos los patrones. La forma de que solo actúen los representantes de la hipótesis elegida es forzar inactividad en el resto de los patrones, y asumir que las TGEs solo utilizarán patrones activos.

Una vez que el simulador termina el proceso de confirmación de la hipótesis, los patrones verificados, entre los descendientes de la hipótesis estudiada, deben quedar inactivos con objeto de que la TGC pueda centrarse en otro área.

Para cada hipótesis **H**, la TGC aplica el siguiente algoritmo de evaluación:

*1) Si **H** ya ha sido evaluada terminar.*
*2) Si Nivel de Activación(**H**) = **No**:*
$\forall$subhipótesis $h \in H$:

Nivel de Activación(h) <- ***No*** *(negar* ***h****).*

3) *Si* ***H*** *tiene Nivel de Activación* ***Posible****:*

   ∀subhipótesis ***h****∈****H****:*

      Evaluar ***h*** *(recursión).*

4) *Si Nivel de Activación(****H****) =* ***Si****:*

   Llevar ***H*** *a la lista de hipótesis candidatas de la TGC.*

Negar una hipótesis consiste en poner su Nivel de Activación a *No activa* y, recursivamente, negar sus propias subhipótesis, con independencia de su valor anterior.

Dada una lista de hipótesis candidatas para su exploración, el proceso de selección se lleva a cabo aplicando la función de interés para cada hipótesis y eligiendo la que arroje un valor mas alto. Este proceso puede realizarse también mediante una base de conocimiento ad hoc de selección.

El proceso de revisión del árbol de hipótesis se realiza cada vez que el simulador ha explorado una hipótesis. Pueden ocurrir dos casos:

1) La tarea de simulación ha generado escenario de comportamiento futuro.
2) La tarea de simulación no ha generado escenario de comportamiento futuro.

Para el primer caso, es necesario revisar la hipótesis estudiada a la baja. Su nivel de activación, antes de entrar en el simulador, era *Activa* pasando -por la revisión- a ser solo *Posible*. La revisión se completa evaluando sus subhipótesis. Estas estarán sin evaluar, salvo que alguna subhipótesis tenga algún padre adicional cuya evaluación haya motivado -a su vez- la evaluación de la subhipótesis. La revisión no se realiza hacia los ascendientes, en el árbol, de la hipótesis explorada ya que estos -forzosamente- deberán tener Nivel de Activación *Posible* o *Activa* (dado el algoritmo de evaluación, al menos debe haber un camino hacia la raíz del árbol de hipótesis con todos los niveles de activación a *Posible*).

La interpretación del proceso de revisión es la siguiente: aunque la tarea de simulación ha generado escenario para la hipótesis seleccionada, no se descarta la posibilidad de generar algún otro para una subhipótesis mas localizada. En este caso, además, la TGC fuerza a que el posible futuro escenario a generar se significativamente diferente de los ya generados. Para ello, y de forma previa a la revisión de la hipótesis explorada, la TGC desactiva los patrones elementales descendientes que hayan sido verificados por el simulador. De esta forma, las hipótesis derivadas de la ya explorada pierden interés para el proceso de simulación y, aunque no se descarta su posterior estudio, la TGC prima la actuación futura del simulador en otras direcciones.

Para el segundo caso se realiza exactamente el mismo tratamiento, si bien la interpretación es diferente: si la tarea de simulación no ha generado escenario de comportamiento para la hipótesis explorada (considerada la más representativa de problemas potenciales) cabe pensar que los posibles problemas están mas localizados. Por tanto, alguna hipótesis descendiente se activará y será posible generar escenario para ella.

La revisión del estado del árbol de hipótesis se completa re-evaluando la lista de hipótesis candidatas del la TGC. Teniendo en cuenta las consideraciones anteriores, el algoritmo de revisión del árbol de hipótesis es como sigue:

1) (Iniciación).

Sea ***H*** *la hipótesis explorada por la tarea de simulación.*

Sea ***LPD*** *la lista de patrones elementales descendientes de* ***H***.

Sea ***FRONTERA*** *la lista actual de hipótesis candidatas de la TGC.*

Sea ***ANTIGUA-FRONTERA*** *<-* ***FRONTERA*** *- {****H****} (la hipótesis explorada no se volverá a explorar).*

FRONTERA *<- {} (se construirá la nueva lista de candidatas).*

∀patrón ***P*** *∈* ***LPD****:*

Si *Verificado(****P****)=****Si*** *(el patrón ha sido verificado en simulación)*

Entonces *Activado(****P****) <-* ***No*** *(se desactiva el patrón).*

*Verificado(****P****) <- {} (el patrón admitirá una futura verificación).*

2) (Revisión)

Revisar la hipótesis explorada ***H***.

∀hipótesis ***h*** *∈* ***ANTIGUA-FRONTERA****:*

Evaluar ***h***.

En ***FRONTERA*** *queda la nueva lista de hipótesis candidatas.*

6) (finalización)

Si ***FRONTERA*** *= {} terminar.*

4) (selección)

Seleccionar la nueva mejor hipótesis para exploración entre las candidatas de ***FRONTERA***.

Sea ***NH*** *la nueva hipótesis seleccionada.*

Sea ***PE*** *la lista de patrones elementales descendientes de* ***NH*** *en el árbol de hipótesis.*

Sea ***N-PE*** *la lista de patrones elementales no descendientes de* ***NH***.

∀patrón ***P*** *∈* ***PE****:*

*Interviene(****P****) <-* ***Si*** *(las restricciones relacionadas con* ***P*** *serán tenidas en cuenta por las TGEs).*

∀patrón ***P*** *∈ N–****PE****:*

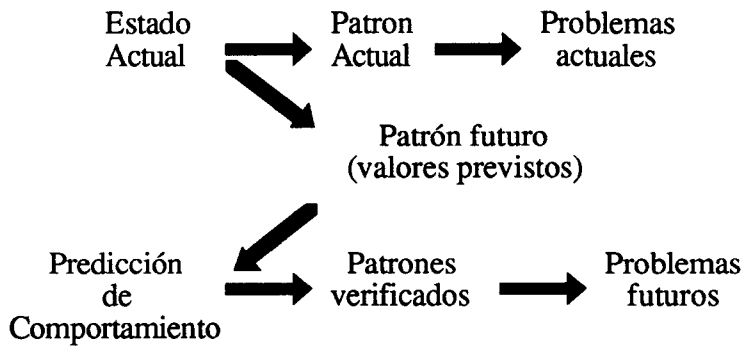
*Interviene(**P**) <- **No** (las restricciones relacionadas con **P** no serán tenidas en cuenta por las TGEs).*

La TGC continúa revisando el árbol de hipótesis, y ordenando simulaciones, en tanto el proceso anterior no detecte lista de hipótesis candidatas vacía.

6.1.1.3. Patrones de comportamiento final.

Un patrón de comportamiento final se define como un objeto que contiene información sobre un esquema tipo de comportamiento de un objeto físico componente del sistema sobre el que se razona. Un patrón define una estado representativo de una situación problemática o de interés, hacia la cual se dirigirá el razonamiento en predicción.

Un patrón se define en términos de las variables de estado del sistema físico. A partir de los valores obtenidos en un patrón se deducirán problemas de diversos tipos. El esquema de inferencia alrededor de un patrón es siempre el mismo:



Como se ve en el esquema de inferencia, un patrón debe contener dos tipos de valoraciones de las variables que controla:

1) Valoración actual, útil para inferir problemas actuales.
2) Valoración prevista, como entrada de las Tareas de Generación de Escenarios y como premisa para inferir problemas en horizontes temporales

A estas se le añade una valoración adicional: Gradiente de Cambio, que es una medida de la distancia entre la Valoración actual del patrón y su Valoración prevista.

Un patrón de comportamiento puede estar en uno de los dos estados siguientes:

1) **Activo**: el patrón está en condiciones de influir durante el proceso predictivo (actuación del modelo profundo).
2) **No Activo**: es el caso contrario.

Paradójicamente, el conocimiento sobre el estado de actividad de un patrón se define sobre el estado **No Activo**, siendo Activo el estado por defecto.

Un patrón define una Superficie de influencia, conjunto de objetos que con él se relacionan. Este conjunto da una medida del tamaño del foco de atención que ha de abarcarse en fase de razonamiento profundo. Para cada patrón, se define una función de interés I que, en general, tomará la forma:

I = I(Gradiente de cambio, Superficie)

Esta función sirve a la TGC para discriminar que conjunciones de patrones deben utilizarse en cada momento para estrechar el pasillo de búsqueda durante el proceso de generación de escenarios. En principio, los patrones con mayor Gradiente de cambio y mayor Superficie serán más interesantes, pero este criterio -en cualquier caso- debe ser revisable por el constructor del modelo.

6.1.2. Tareas de Generación de Escenarios (TGEs).

Las Tareas de Generación de Escenarios (TGEs) tienen por objeto representar el conocimiento necesario para filtrar las influencias de unos objetos físicos sobre otros. Su misión es, por tanto, generar los escenarios de entrada para un tipo dada de Tareas Básicas de Simulación (TBSs) a partir de los escenarios de salida generados por TBSs antecedentes en el sistema de flujo, reduciendo el espacio de comportamientos a investigar. El estrechamiento del pasillo de búsqueda se produce según dos criterios principales:

1) Verosimilitud de las combinaciones de valores aportadas por las TBSs entrantes.
2) Pertinencia de (estudiar) las combinaciones de valores entrantes de acuerdo con los patrones de comportamiento final preestablecidos globalmente por la Tarea General de Control (TGC); es decir, criterio de relevancia por objetivos.

Se describe a continuación la representación del conocimiento y la estructura de la tarea.

6.1.2.1. Representación del conocimiento.

La estructura de representación se define mediante un marco con varias categorías de atributos, y dos bases de conocimiento para representar los criterios de filtrado.

En los siguientes subapartados se describen cada una de las categorías de atributos y las bases de conocimiento.

Atributos básicos para la definición de un escenario.

Los atributos básicos son las variables descriptivas del comportamiento compartidas por diferentes TBSs en niveles consecutivos de simulación. Un escenario se define, en consecuencia, como un conjunto de asignaciones individuales de valores a estos atributos.

Antes de producirse la llamada a una TGE, cada atributo de definición del escenario tendrá asignado el subdominio de posibles valores futuros. Dicha asignación se habrá producido por la acción individual de cada TBS entrante a la TGE. La acción de la TGE restringirá esos dominios conservando un valor para cada atributo básico, dando lugar al escenario de salida.

Por ejemplo, en el dominio de la Predicción de Avenidas en Cuencas Hidrográficas, los atributos básicos podrían ser los caudales de respuesta de superficies receptoras de lluvia que actúan como caudales entrantes en un barranco. Las superficies receptoras se representarían como TBSs entrantes y el barranco como TBS saliente. La TGE actuaría como filtro intermedio eligiendo una combinación (escenario) de caudales respuesta (entre las distintas hipótesis obtenidas en cada superficie receptora de lluvia) para actuar en los puntos de drenaje del barranco.

Debe hacerse notar que los atributos básicos, al representar los comportamientos en el tiempo producidos por las TBSs entrantes, no contendrán una gama de valores instantáneos, sino una gama de series de valores en un horizonte temporal. Así, en el ejemplo anterior los caudales de respuesta de las superficies receptoras serían, en el caso general, hidrogramas alternativos de caudal, como se refleja en la figura 6c; es decir, series temporales de caudal alternativas, convertidas a volúmenes acumulados en distintos horizontes temporales y en escalas cualitativas prefijadas.

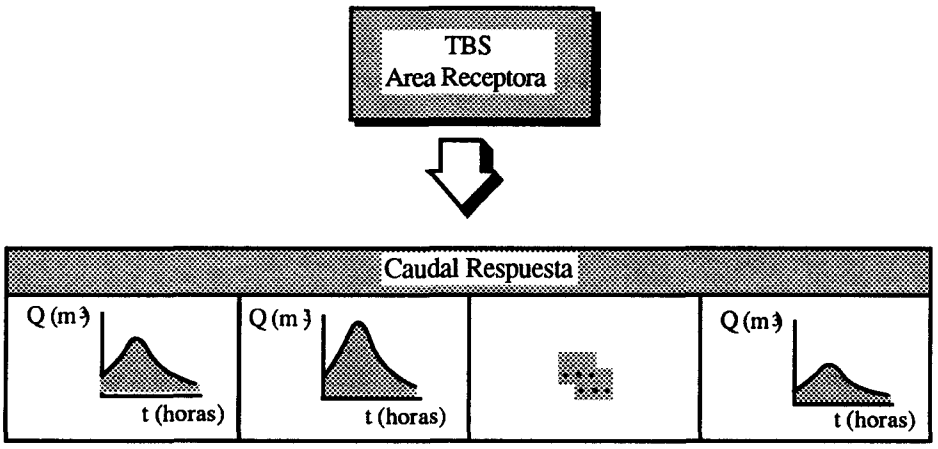


Figura 6c: Respuestas alternativas de una TBS.

La figura 6d ilustra la transformación que la TGE impone en los atributos básicos, reduciendo la gama inicial de valores a un único valor representante por atributo. Los atributos básicos de definición del escenario, al ser las variables compartidas por TBSs en niveles consecutivos de simulación, definen además la topología de la red de objetos físicos, que sirve de guía para el proceso de la Tarea de Simulación Cualitativa.

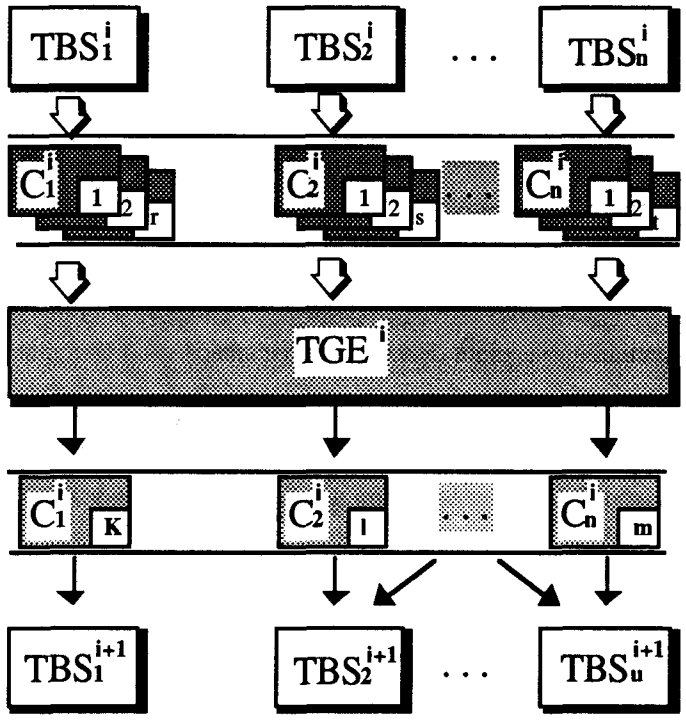


Figura 6d: Filtrado de TGE sobre atributos básicos.

Patrones de comportamiento final.

Un patrón se define por un valor que describe el nivel de cambio de una variable. Los patrones sirven como elementos de filtrado para encontrar los escenarios relevantes según los objetivos de búsqueda impuestos por la TGC.

El conjunto de TBSs entrantes a la TGE habrá generado una gama de comportamientos, alguno de los cuales -o todos ellos- estarán representados en variables relacionadas con un patrón entre los activos. A partir de la definición de los patrones el objetivo de la simulación cualitativa, en su conjunto, es explorar y decidir si es factible que las variables comportamiento muestren una dinámica similar a la establecida en los patrones. De esta forma, los patrones de comportamiento definen realmente **bandas de aceptabilidad** para los comportamientos producidos por las TBSs.

En la figura 6e se muestra la actuación de un patrón sobre una variable V con tres comportamientos C1, C2 y C3. Unicamente el comportamiento C2 sería considerado factible por la TGE.

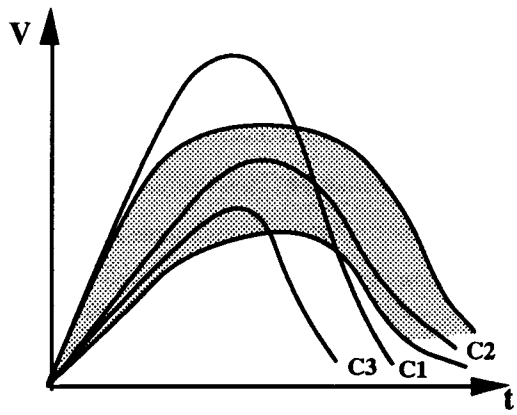


Figura 6e: Influencia de un patrón sobre una variable.

Debe resaltarse que las regiones de factibilidad definidas por los patrones no contienen los comportamientos considerados más probables sino aquellos que, de ser factibles, son relevantes para la toma de decisiones. En este contexto, la misión de la simulación cualitativa no es predecir (o sea, acertar) el comportamiento futuro del sistema a partir de la predicción de acciones externas. Por el contrario, su misión es identificar, entre los comportamientos posibles, aquellos que pueden derivar en problemas. De hecho, el caso normal será que el sistema real, objeto de modelización, presente sus comportamientos fuera de los patrones; es decir, que se ajusten a lo que sería un patrón de normalidad. Dichos comportamientos son, paradójicamente, irrelevantes para la toma de decisiones ante problemas.

En consecuencia, ante una situación de normalidad, la TGE decidirá la infactibilidad de los patrones propuestos por la TGC y, de hecho, detendrá el proceso de simulación. La Tarea de Simulación Cualitativa no generará ningún escenario global de comportamiento, lo cual será interpretado por la TGC de dos formas:

1) Estado de normalidad en todo el sistema.
2) Los comportamientos problemáticos abarcan un área más reducida de la considerada.

En el primer caso, La TGC detendrá el proceso global, no reanudandose este hasta una nueva lectura del Estado Inicial. En el segundo caso, la TGC reducirá el foco de atención de la simulación estableciendo nuevos patrones de comportamiento final a explorar que serán:

1) Más optimistas desde el punto de vista del sistema global, pues se reducirán el número de problemas potenciales a considerar.
2) Más precisos, pues involucrarán a un número menor de componentes del sistema.

Patrón referencial de evolución.

El patrón referencial se define como una asignación concreta de valores a los atributos básicos. Se concibe para actuar de referencia en el proceso de generación del escenario por parte de la TGE. Esto quiere decir que, entre los diferentes escenarios factibles y verosímiles, la TGE elegirá como respuesta aquel mas cercano al patrón referencial. Para ello se define como "distancia" entre un escenario y el patrón referencial al número de variables que toman distinto valor en uno y otro conjunto.

La asignación del patrón referencial habrá sido deducida en las TBSs entrantes según uno de los dos criterios siguientes:

1) Cuando exista una tendencia reciente conocida para los atributos básicos, el patrón referencial será dicha tendencia. Este criterio será aplicable en aquellos dominios para los cuales sea posible asumir la hipótesis inercial de [Shoham 88] de que un sistema físico tiende a mantener la evolución actual, de manera que cambios menores respecto a la tendencia son más probables que cambios importantes.
2) Cuando no sea posible identificar la tendencia reciente (o no sea asumible la hipótesis inercial) el patrón referencial puede definirse de forma alternativa según criterios estratégicos (por ejemplo, la evolución más pesimista, una evolución media, etc). En este caso, el patrón referencial no actúa necesariamente para seleccionar el escenario considerado más verosímil, sino aquel más interesante según la estrategia general establecida.

En principio puede parecer excesivamente restrictivo que la TGE genere un único escenario (el más cercano al patrón de referencia). De hecho, cabría plantear la generación de varios escenarios; por ejemplo, los N mas cercanos al patrón referencial, siendo N un número prefijado o bien deducido en la TGC alrededor de cada hipótesis de exploración. Ello implicaría un proceso de búsqueda para la Tarea de Simulación Cualitativa, dando lugar a un árbol de búsqueda como el de la figura 6f.

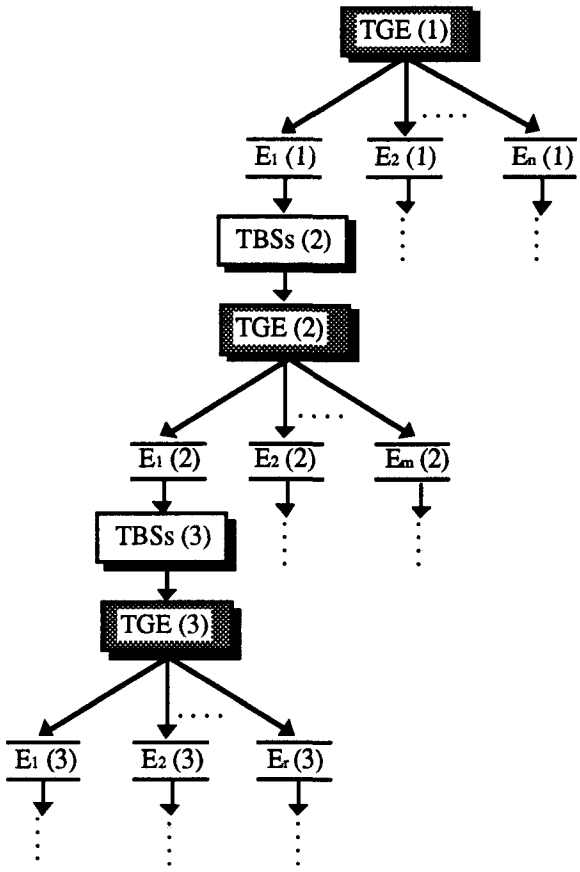


Figura 6f: Arbol de búsqueda potencial en una simulación.

Para cada escenario producido en el árbol de búsqueda, se establecería una linea separada de razonamiento. La Tarea de Simulación Cualitativa realizaría el correspondiente "backtracking" cuando una TGE decidiera infactibilidad, retrocediendo hasta la TGE en el nivel inmediatamente anterior de la simulación y continuando el proceso a partir del siguiente escenario. El proceso de búsqueda podría ser con estrategia preferente en profundidad o amplitud y, en cualquier caso, exhaustivo o limitado a la generación de un número M de escenarios globales de comportamiento. Análogamente al caso de la generación de escenarios en cada TGE, el número M sería prefijado o bien deducido en la TGC en base a la hipótesis de exploración en curso.

Este planteamiento presenta aspectos problemáticos. Efectivamente, generar varios escenarios de salida en la TGE puede dar lugar fácilmente a que la Tarea de Simulación deduzca globalmente comportamientos que no sean

significativamente diferentes, en particular si los escenarios están ordenados por proximidad al patrón referencial. Esto implica realizar esfuerzos inútiles de simulación.

Por otra parte, establecer un criterio diferente de ordenación significaría rechazar escenarios considerados, a priori, más probables o -en cualquier caso- romper el criterio estratégico establecido en las TGEs.

Realizar un "backtracking" más inteligente a nivel de la Tarea de Simulación seleccionando, en situaciones de "backtracking", escenarios no consecutivos y evitando, en consecuencia, la generación de comportamientos globales similares podría ser la alternativa adecuada. Ello significaría, sin embargo, realizar nuevamente procesos inútiles -esta vez en las TGEs- pues se habrían generado escenarios que luego no se utilizan. Además, resultaría muy difícil definir criterios para establecer la "distancia" adecuada entre selecciones.

Finalmente, cabría plantear que las propias TGEs tomen el control del proceso de búsqueda, de forma que generen un primer escenario y, solo en situaciones de "backtracking", vayan produciendo escenarios alternativos. Ello exigiría, inevitablemente, que las TGEs razonaran alrededor de los comportamientos globales ya producidos y de los patrones actuales de comportamiento, estableciendo nuevos patrones. Esto podría provocar inconsistencias entre los escenarios generados a distintos niveles de la misma linea de razonamiento en el árbol de búsqueda. Para evitarlo, la Tarea de Simulación, ante un punto de "backtracking" en una TGE y con los nuevos patrones de comportamiento establecidos, tendría que revisar la linea de razonamiento completa desde la raíz del árbol de búsqueda hasta el punto de retroceso.

La última solución apuntada es factible, pero a costa de modificar la funcionalidad de la TGE que pasaría a entender no solamente de las formas de influir de unos objetos físicos sobre otros, sino también de como definir las estrategias globales. Además, la última forma de "backtracking" mencionada es precisamente la que se realiza en la TGC.

En consecuencia, la generación de un único escenario en cada TGE no es un criterio tan restrictivo como pueda parecer a simple vista, pues el árbol de búsqueda se produce a nivel global, controlado por la TGC. Esto tiene la enorme ventaja de evitar la creación de lineas de razonamiento que, derivando en comportamientos redundantes, sean por tanto inútiles.

Base de conocimiento de Pertinencia.

Esta base contiene conocimiento para efectuar la regresión de una conjunción de patrones de comportamientos final suministrada por la TGC. Esta base es, pues, el vehículo para implementar razonamiento abductivo desde

los problemas potenciales hacia posibles valores de las variables de estado que son su causa; en definitiva, para realizar simulación guiada por objetivos.

El conocimiento se representa, básicamente, por medio de restricciones que relacionan valores de patrones de comportamiento con valores posibles de cambios en atributos básicos e infiriendo, por tanto, valores necesarios en estos atributos para que se cumpla la conjunción de patrones impuesta.

El conocimiento representado es una versión superficial del conocimiento que representa al comportamiento. Se utiliza, en consecuencia, para obtener condiciones necesarias en las predicciones realizadas por las TBSs entrantes, de manera que si dichas predicciones no cumplen las condiciones impuestas se detendrá la simulación, pues esta llevaría, según los patrones propuestos, a comportamientos irrelevantes o bien pesimistas en exceso.

Notese como el hecho de que una determinada hipótesis de exploración (conjunción de patrones) sea declarada infactible por la TGE no significa abandonar dicha hipótesis en su totalidad. Como ya se apuntó en el apartado 6.1, a raíz de la infactibilidad de una hipótesis, la TGC propondrá nuevas hipótesis de exploración, entre las cuales existirá -forzosamente- alguna que constituya una versión relajada de la hipótesis rechazada por la TGE (es decir, una conjunción más débil de patrones) lo que puede dar lugar, posteriormente, a escenarios de comportamiento global más optimistas.

Los elementos de conocimiento de la Base de Pertinencia pueden obtenerse, en general, a partir de experiencia previa (experiencia artificial con modelos de simulación cuantitativa o -preferiblemente- experiencia natural cuando exista un historial de comportamientos del sistema modelizado suficientemente grande).

Aunque dependerá de cada dominio, en el caso general la Base de Pertinencia se descompondrá en dos subbases:

1) Base de identificación de restricciones activas. El conocimiento de esta base se puede definir mediante reglas que deduzcan, a partir de los patrones activos planteados, valores posibles de variables intermedias para actuar como terminos independientes de las restricciones que, finalmente, aplican los criterios de pertinencia.

   Este tipo de bases serán necesarias cuando no sea posible relacionar directamente los patrones con los atributos básicos. Por ejemplo, en el caso de estudio propuesto en el Anejo A (predicción de comportamiento en Cuencas Hidrográficas) los patrones de lluvia se establecen por zonas geográficas donde se supone un comportamiento meteorológico uniforme, en tanto que las predicciones de lluvia de las diferentes zonas se relacionan (criterio de pertinencia) por umbrales máximos y mínimos de volúmenes de lluvia a ser alcanzados por agregaciones de dichas zonas. Es necesario, por tanto, efectuar un paso de inferencia previo que, a partir de los patrones de lluvia activos, identifique esos

umbrales de lluvia e, indirectamente, qué restricciones deben tenerse en cuenta para aplicar los criterios de pertinencia según objetivos.

2) Base de inferencia de subdominios pertinentes de los atributos básicos. Contiene el conocimiento que define los criterios de pertinencia propiamente dichos, y se representa mediante restricciones cualitativas que relacionan los atributos básicos de la TGE con los términos independientes deducidos por la base anterior. propio

Base de conocimiento de Verosimilitud.

Esta base contiene conocimiento para decidir qué combinaciones, entre las predicciones de comportamientos generados por las TBSs entrantes, resultan verosímiles. El conocimiento se representa mediante restricciones que modelizan propiedades de la evolución temporal y espacial de atributos básicos; es decir, que establecen cuando dicha evolución se considera consistente.

Para una variable dada, el primer criterio servirá para imponer umbrales de crecimiento y/o decrecimiento en un horizonte temporal. El segundo criterio servirá para representar conocimiento de correlación entre variables de distintos objetos; por ejemplo, que dos variables deben tener un comportamiento análogo). Finalmente, ambos criterios deben poder ser aplicados de forma simultanea; por ejemplo, si se desea limitar el crecimiento conjunto de una serie de variables.

Análogamente a la Base de Pertinencia, dependiendo del dominio esta base se podrá descomponer en diferentes subbases de reglas y restricciones, si la estrategia para aplicar los criterios de verosimilitud así lo requiere.

Discusión sobre los criterios de filtrado.

A efectos de que la TGE pueda decidir la factibilidad de una conjunción de patrones, y generar un escenario en consecuencia, es necesario que los dos criterios anteriores, pertinencia según objetivos y verosimilitud, se satisfagan simultáneamente.

Efectivamente, si los sistemas de restricciones de pertinencia y verosimilitud comparten alguna variable (lo normal será que compartan los atributos básicos) el efecto de la propagación de restricciones sobre los dominios de las variables compartidas puede determinar su insatisfacibilidad en casos en los que, cada sistema por separado, permitiría su existencia.

Por ejemplo, supóngase dos sistemas de restricciones A y B con una variable compartida V que toma valores en el dominio $\{v_1, v_2, ..., v_n\}$. La situación se ilustra en la figura 6g, siendo:

$$A(V) \equiv \{v_i, ..., v_k\}$$

el subdominio para V resultante de aplicar el sistema de restricciones A y:

$$B(V) \equiv \{v_r, ..., v_t\}$$

el subdominio para V resultante de aplicar el sistema de restricciones B.

Se tendrán cuatro casos posibles:

1) $A(V) \equiv B(V)$. En este caso A(V) -o B(V)- es el subdominio final para V.
2) $B(V) \supset A(V)$. Será necesario calcular nuevamente B(V) tomando A(V) como dominio de partida para V. Esto es así porque puede existir alguna restricción que reduzca el dominio de V -o de variables adyacentes[1] a V- en B.
3) $A(V) \supset B(V)$. Es el caso contrario al 2) y con idénticas implicaciones.
4) $A(V) \cap B(V) \neq \{\emptyset\}$ y sin cumplirse ninguno de los casos anteriores. Será necesario calcular nuevamente A(V) y B(V) dado que pueden existir restricciones que reduzcan el dominio de V -o de variables adyacentes a V- en A y B.

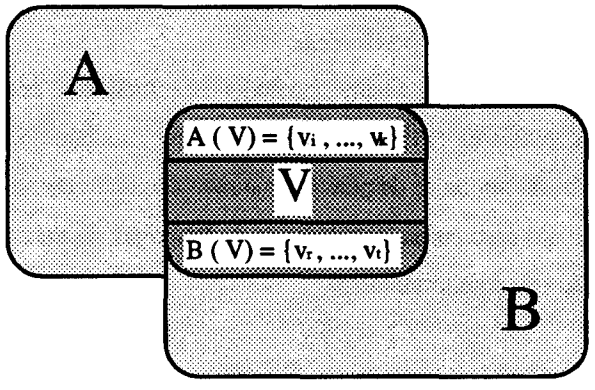


Figura 6g: Efecto de dos sistema de restricciones sobre una variable.

Y el esquema anterior se plantea cada vez que se aplique A o B sobre V.

---

1 *Dos variables son adyacentes si están relacionadas por alguna restricción.*

Realmente, cualquier algoritmo de satisfacción de restricciones de la literatura recoge el problema descrito ya que siempre se cumple:

$$A(V) \cap B(V) \supseteq [A + B](V)$$

siendo [A + B] el sistema de restricciones conjunto.

Por consiguiente, desde el punto de vista práctico de obtener rápidamente el subdominio final de los atributos básicos de la TGE, la opción más eficiente sería satisfacer una única vez y conjuntamente el sistema de restricciones suma de los criterios de pertinencia y verosimilitud. No es esta, sin embargo, la opción elegida. La ejecución conjunta de ambos sistemas permitiría deducir simultáneamente la utilidad y la consistencia (a nivel de dominios) de las combinaciones de valores de las predicciones de entrada, pero a costa de reducir la calidad explicativa de la TGE. Supuesta una situación de infactibilidad, se desconocería su origen, aspecto importante, en particular respecto de la depuración de modelos.

De la discusión anterior se deriva, por tanto, la necesidad de establecer una base de síntesis, unión de las dos bases de restricciones de pertinencia y verosimilitud, que debe aplicarse con posterioridad a ambas. Evidentemente, cabe la posibilidad de que sea la base de síntesis la que decida la infactibilidad de una hipótesis de exploración; es decir, que los sistemas de pertinencia y verosimilitud resulten satisfacibles por separado pero no conjuntamente.

En general, la infactibilidad de los dominios de partida se explicará por una de las razones siguientes:

1) Los dominios de partida no son pertinentes según los objetivos propuestos.
2) Los dominios de partida no son verosímiles.
3) Las dos razones anteriores conjuntamente, sin poder especificar cual es la causa de la infactibilidad.
4) Hay una inconsistencia en el modelo; es decir, existe un conjunto de restricciones insatisfacible con independencia de los dominios de partida (por ejemplo conjuntos del tipo {X=Y, X≠Y} o {X<Y, Y<Z, Z<X}, etc). Este último caso puede detectarse de forma previa sin más que ejecutar el sistema con cada variable asignada con su dominio total.

6.1.2.2. Estructura de la tarea.

El motor de inferencia de la TGE, para cubrir el objetivo de proponer un escenario de predicciones, opera en dos pasos:

1) Un paso de restricción de dominios de las predicciones de comportamientos entrantes.

2) Un paso de generación de un escenario que cumple las condiciones impuestas de verosimilitud y pertinencia para los objetivos.

La estructura general de la tarea se muestra en la figura 6h. La estrategia se estructura en dos lineas de razonamiento paralelas para aplicar cada uno de los criterios de filtrado establecidos. En este punto cabría considerar criterios adicionales (si hubiere) que se situarían en batería con los dos mencionados. Las lineas de razonamiento para filtrado así establecidas confluyen en un paso de síntesis que da lugar a los comportamientos de los cuales crear el escenario de salida.

La estrategia completa se realiza según los siguientes pasos:

1) Paso de restricción de dominios.
   1.1) Regresión de patrones. Utilizando la base de conocimiento de pertinencia (base SSR1), y a partir de las predicciones de comportamientos y la conjunción de patrones de comportamiento final, se realiza un proceso de satisfacción de restricciones tipo Waltz [Winston 84], obteniendose los comportamientos relevantes (C_r) para los objetivos. Como ya se señaló anteriormente, cabe considerar la existencia de bases adicionales, en la linea de razonamiento, para la deducción de dominios iniciales de variables intermedias. En ese caso, dichas bases se evaluarían de forma previa al sistema de restricciones.
   1.2) Evaluación de la consistencia espacial y temporal. Utilizando la base de conocimiento de verosimilitud (base SSR2), y a partir de las predicciones de comportamientos, se realiza un proceso de satisfacción de restricciones, obteniendose los comportamientos consistentes (C_c). Análogamente a la regresión de patrones, bases adicionales se aplicarían préviamente.

   Si de cualquiera de los dos pasos anteriores se obtiene un conjunto de comportamientos vacío, se decide que la conjunción de patrones propuesta no es factible, bien porque los comportamientos de entrada no son relevantes (linea de regresión) o bien porque no son consistentes (linea de verosimilitud).

   1.3) Examinando la intersección de los dominios propuestos por ambas lineas de razonamiento $\{C_r \cap C_c\}$ puede obtenerse igualmente el dominio vacío. En caso contrario se aplica a dicha intersección el sistema de restricciones conjunto $\{SSR1 \cup SSR2\}$ obteniendose un conjunto de comportamientos relevantes y consistentes. Si el conjunto es vacío se decide la infactibilidad del patrón propuesto.

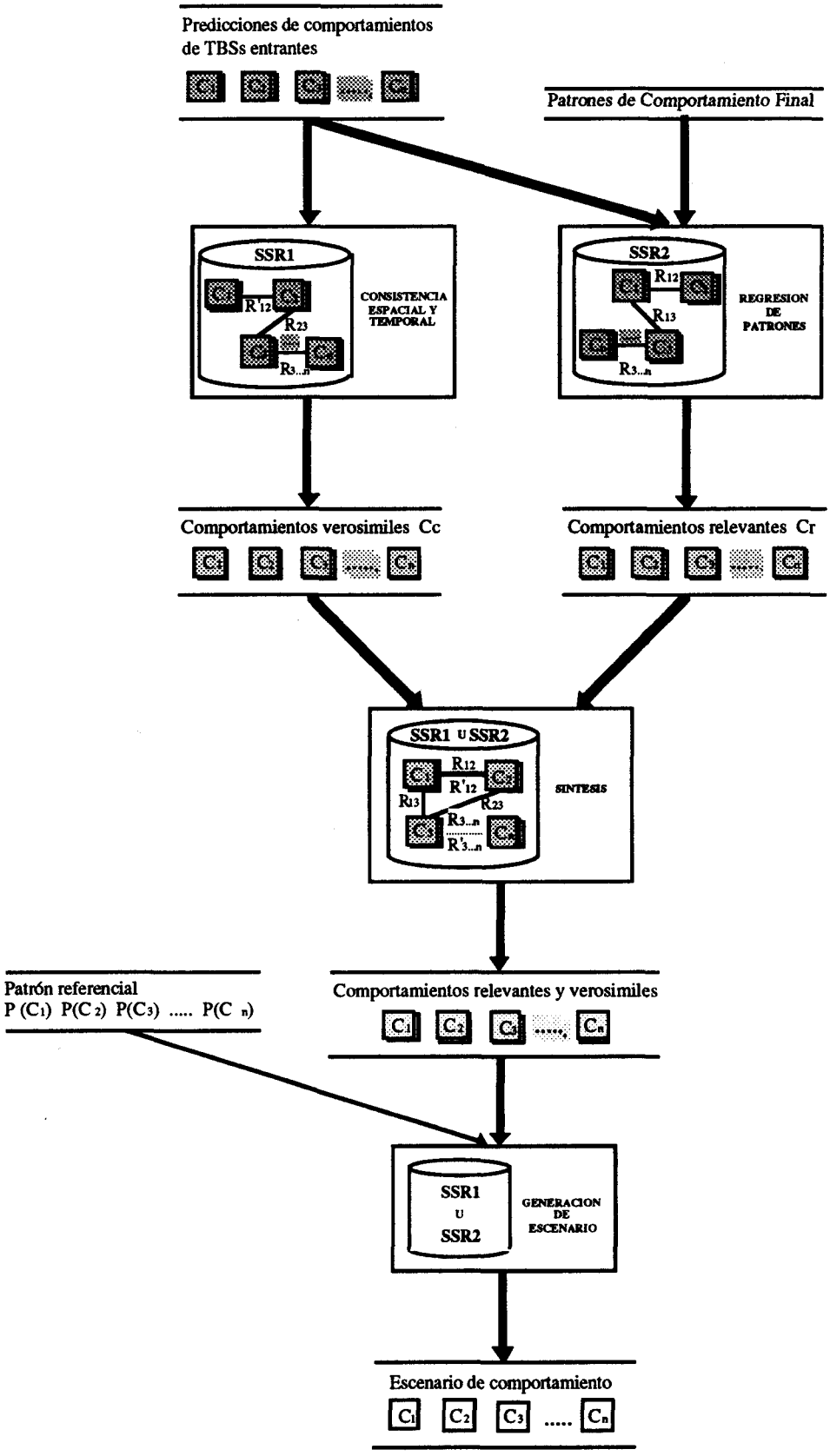


Figura 6h: Estructura de la Tarea de Generación de Escenarios.

2) Paso de generación de escenario.

   Usando los dominios resultantes de 1), y apoyandose nuevamente en el conjunto de restricciones {SSR1 ∪ SSR2} se exploran las diferentes combinaciones de los comportamientos préviamente filtrados. Para este paso se puede utilizar un algoritmo de generación y test de combinaciones [Izquierdo 92], si bien con el criterio heurístico adicional de producir las combinaciones en orden de proximidad al patrón de referencia.

El proceso finaliza cuando se obtiene una primera interpretación. Los valores individuales asignados a los atributos básicos, dentro de la interpretación, conforman el escenario de comportamientos propuesto por la TGE, para actuar de solicitación de las TBSs en el siguiente nivel de simulación.

6.2. TAREAS GENERICAS PARA SIMULACION DEL COMPORTAMIENTO: TAREAS BASICAS DE SIMULACION.

Las Tareas Básicas de Simulación (TBSs) representan la respuesta de los diferentes objetos físicos que integran el sistema de flujo.

El conocimiento interno de una TBS es un modelo cualitativo del comportamiento del objeto, relacionando el escenario de acciones externas al mismo con sus posibles respuestas. Por ejemplo, en el dominio de la predicción de inundaciones en cuencas hidrográficas, podrían definirse como TBSs las Areas Receptoras, para modelizar el comportamiento de una superficie sometida a la acción de la lluvia, produciendo como respuesta caudales de entrada para una red de transporte (que también se representaría mediante una TBS).

En conjunto, el objetivo de una TBS es generar predicciones del comportamiento de un componente del sistema en un cierto horizonte temporal. Se trata, por tanto, de modelizar las transiciones de estado que el objeto sufre a lo largo del tiempo, esto es [Cuena, Salmerón 89]:

> "Dado un estado inicial y una predicción de acciones externas que pueden actuar sobre el objeto en un futuro cercano, deducir posibles estados futuros a lo largo del tiempo".

La forma de razonamiento debe constituir una abstracción de los métodos utilizados por los profesionales para evaluar los comportamientos, sustituyendo los cálculos basados en medidas numéricas por relaciones entre intervalos de valores o escalas cualitativas. En ingeniería existe una tradición importante en el uso de modelos matemáticos de simulación, encuadrados en la forma de operar de las personas en dos pasos:

1) Razonamiento desde los datos de características físicas y de estado para establecer las condiciones en las que debe aplicarse el modelo.

2) Aplicación del modelo al conjunto de opciones para obtener los comportamientos previsibles.

La TBS constituye una versión cualitativa de las técnicas de simulación usadas habitualmente. Según el tipo de modelo de simulación cabe diferenciar dos casos:

1) Elementos con modelo basado en transiciones explícitas, cuyo comportamiento puede formularse con bases de conocimiento con el formato:

$$E(t),\ A(t,\ t+\Delta t) \Longrightarrow E(t+\Delta t') \qquad (1)$$

siendo E(t) el estado del objeto en t y A(t, t+Δt) el conjunto de acciones externas durante el periodo [t, t+Δt].

Para los casos en que es posible obtener relaciones cognitivas de este tipo, el procedimiento de interpretación permite obtener un árbol de "envisionment" con etiquetado temporal de los vértices aplicando en bucle, durante el periodo a simular, la base de conocimiento tipo (1). Esta aplicación iterativa puede hacerse con restricciones de control que eviten la ejecución en secuencia de determinadas reglas (por ejemplo, podría prohibirse encadenamientos solo de aplicaciones de reglas que den lugar a transiciones pesimistas -u optimistas- podando dichas ramas del árbol de "envisionment").

Bajo esta forma de operar, las reglas anteriores enumeran las relaciones entre los valores de las variables premisa y las variables conclusión, en los distintos horizontes temporales. Por ejemplo, cabría definir una gama de comportamientos (series de valores en el tiempo) etiquetados a, b, c, de manera que pudieran formarse reglas del tipo:

SI estado del suelo **ES** saturado **Y**
acción externa de la lluvia **ES** alta
ENTONCES tipo de comportamiento **ES** a

Esta forma de representación es posible cuando las personas expertas pueden formular reglas como las anteriores, o bien es posible plantear su adquisición mediante técnicas de aprendizaje.

2) Objetos con modelo de transición implícita. En este caso la forma de evolución en el tiempo del objeto depende de las soluciones de un sistema de confluencias que condiciona los signos de evolución de las variables de estado. Este paradigma se corresponde con los modelos [Kuipers 86] y [DeKleer, Brown 84a, 84b] en los que se obtiene el árbol de "envisionment" vía un proceso iterativo de satisfacción de restricciones.

   Esta aproximación se producirá para aquellos objetos para los cuales las personas expertas no sean capaces de formular formatos de reglas superficiales, o bien se difícil su síntesis automática.

En ambos casos, el conocimiento está formado por los criterios de transición de un estado -en el momento inicial- a posibles nuevos estados en momentos sucesivos, dando lugar a un árbol de posibles opciones de evolución, en contraposición a los modelos clásicos numéricos, en los que se fija un proceso determinista de transición de un estado a otro. El razonamiento cualitativo, al amparo de cualquiera de las técnicas descritas, propone diversas posibilidades de transición de forma que, a lo largo del tiempo, permite obtener un árbol cuyo vértice corresponde a un estado inicial y tiene un conjunto de vértices siguientes (al cabo de un determinado tiempo, conocido o no) que, a su vez, tienen otros posibles siguientes, etc.

En el caso general, este árbol de posibles estados es lo que puede afirmarse del comportamiento de un objeto físico complejo sobre el que, por razones económicas o porque no hay tiempo para ello, nunca se tiene información completa.

Las TBS supone, por tanto, una forma de encapsular el razonamiento cualitativo sobre un objeto ya que es capaz de incorporar el concepto de estado del componente, las influencias que sobre él operan, así como de los criterios mencionados de transición entre estados.

Se describen a continuación la estructura de representación y la estructura general de la tarea.

6.2.1. Representación del conocimiento.

De forma análoga a la TGE, la estructura de representación para las TBSs se define mediante un marco con varias categorías de atributos y un conjunto de bases de conocimiento. En los subapartados siguientes se detallan ambos aspectos.

6.2.1.1. Caracterización física.

Se define mediante atributos que representan las características físicas primarias y permanentes del componente. Su valor, por tanto, es constante entre diferentes periodos de simulación. Ejemplos típicos de este tipo de atributo serían:

- Superficie (Km^2) de un Area Receptora de Lluvia.
- Pendiente (%) de un Tramo de Cauce.
- Tiempo de respuesta (ηs) de un componente electrónico.
- PH (1-14) de un compuesto químico.
- Etc.

Los valores de este tipo de atributo deben ser asignados por el profesional durante la creación del modelo.

Dentro de esta categoría cabe distinguir, como subconjunto, aquellos atributos que, siendo su valor una constante intrínseca al objeto físico, su asignación requiere razonamiento. Es decir, son características permanentes deducibles de los atributos primarios para su uso en simulación, pero no son, sin embargo, de fácil adquisición directa. Por ejemplo, la forma de la respuesta de un Area Receptora a la acción de la lluvia (lo que se denomina

hidrograma unitario) es una característica propia (e invariante) de cada área [Chow et.al 87], deducible de atributos primarios como la forma de la cuenca, su tipo de vegetación, su pendiente media, densidad de drenaje, etc.

Si bien la asignación de este subgrupo de atributos requiere razonamiento, una vez realizado este los valores asignados pasan a ser constantes. Por tanto, este razonamiento debe realizarse en tiempo de creación de cada instancia de TBS, de manera que solo se efectúe una vez.

6.2.1.2. Entradas.

Son atributos para la representación de solicitaciones al objeto físico. Realmente, serán predicciones, en un cierto horizonte temporal, de acciones externas que provienen como salidas de objetos antecedentes en el sistema de flujo (es decir, producidas por otras TBSs) o bien directamente del entorno en el que evoluciona el sistema. Ejemplos de atributos de entrada serían:

- Un pluviograma que actúa sobre un Area Receptora de Lluvia.
- Una intensidad de corriente actuando sobre un componente electrónico a lo largo de un periodo de tiempo.
- Una presión de entrada a un regulador de presión.
- Etc.

La evolución en el tiempo de una variable de entrada se entenderá representada de forma discreta; es decir, como secuencia de valores (o intervalos de valores) correspondientes a intervalos de tiempo de longitud prefijada. Por ejemplo, una previsión de lluvia como entrada de un Area Receptora se podría representar como en la figura 6i, con la intensidad de lluvia expresada en milímetros a intervalos de tiempo de cinco minutos.

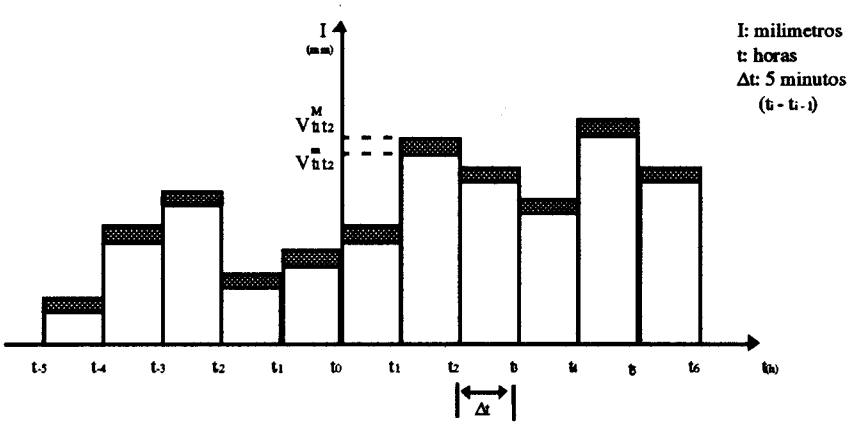


Figura 6i: Pluviograma de entrada para una TBS de Area Receptora de lluvia.

siendo:

I: intensidad de lluvia expresada en milímetros.

$V^{M}_{t_1 t_2}$: intensidad máxima de lluvia en $(t_1, t_2]$.

$V^{m}_{t_1 t_2}$: intensidad mínima de lluvia en $(t_1, t_2]$.

t: tiempo expresado en horas.

Δt: $t_i - t_{i-1}$: intervalo de tiempo entre previsiones.

Nótese como, si bien en general las variables de entrada tendrán como valor evoluciones previsibles en el tiempo, debe contemplarse también la posibilidad de representar evoluciones históricas. Esto será necesario cuando la TBS modelice comportamientos de tipo reactivo, para los cuales se requiera una muestra significativa de valores en el pasado, con objeto de deducir estados iniciales.

6.2.1.3. Definición del Estado.

Son los atributos que caracterizan el estado del objeto físico. Globalmente, dichas variables definen su respuesta. Típicamente, un subconjunto de las variables de estado actuarán como variables de salida para ser acciones externas de otros objetos, en el contexto del sistema de flujo, quedando el resto como variables internas.

En objetos físicos complejos, cuyo comportamiento sea de tipo secuencial (no combinacional) el esquema funcional de definición del estado en un tiempo t puede verse como una generalización del formato superficial presentado anteriormente; es decir:

$$S(t) = E(t).$$

$$E(t) = g(A(t-n\Delta t, t), E(t-n\Delta t), E(t-(n-1)\Delta t), ..., E(t-\Delta t)).$$

siendo:

S(t) : salida del objeto en t.

$A(t, t+\Delta t)$: acciones externas en el intervalo $[t, t+\Delta t]$.

E(t) : estado del objeto en t.

Muchos de los modelos utilizados típicamente en ingeniería tienen la forma general anterior. Siendo así, la TBS debe suministrar capacidad de representación e inferencia para:

1) Deducir características permanentes a partir de los atributos de caracterización física.
2) Deducir estados iniciales.
3) Realizar un paso de simulación; es decir, deducir valores de las variables de estado a partir de ellas mismas, en intervalos anteriores de tiempo, y de las acciones externas.
4) Deducir los atributos de salida a partir de las variables de estado.

Dado que, en general, se manejan escalas cualitativas para cada uno de los tipos de variables mencionados, cada uno de los pasos de inferencia anteriores se podrá producir de forma no determinista. En particular, la realización del paso 3) constituye de hecho una tarea elemental interna de la TBS que, aplicada en bucle para el conjunto de acciones externas, definirá el árbol de "envisionment" que será el reflejo de los posibles comportamientos del objeto. En la figura 6j se muestra la forma de un árbol de transición entre estados (se asume un sistema reactivo de longitud unidad).

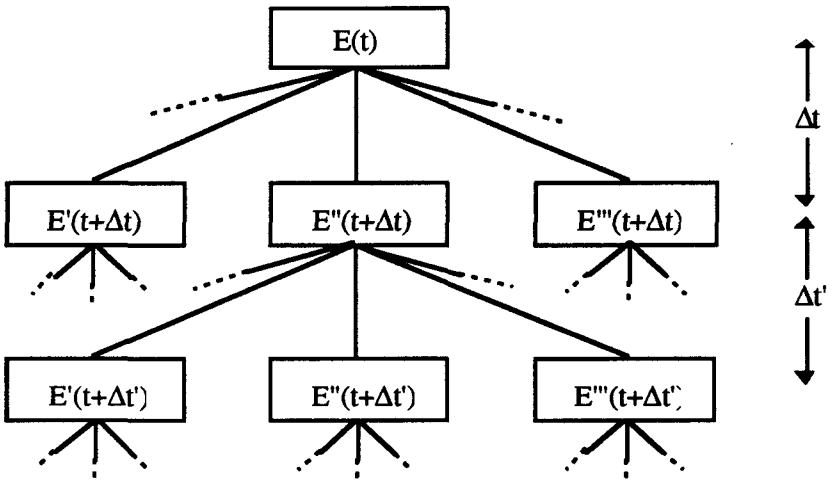


Figura 6j: Arbol de transición de estados.

Del indeterminismo con que, en general, se producirá el paso de simulación se deriva el hecho de que en el árbol de estados anterior se tendrá explosión combinatoria. Es necesario, por tanto, disponer de conocimiento de síntesis que pode ramas del árbol. Por ejemplo, se pueden ordenar las transiciones entre estados por orden de pesimismo y podar aquellos caminos que se consideren excesivamente pesimistas -u optimistas-.

Para las necesidades de representación e inferencia mencionadas, la TBS incorpora una colección de bases de conocimiento. Cada base se podrá formular mediante:

1) Reglas de inferencia. Estas bases es podrán estructurar según una colección de objetivos a satisfacer durante su aplicación.
2) Sistemas de satisfacción de restricciones.
3) Módulo funcional (para aquellos objetivos que admitan un cálculo determinista).

Dichas bases se enumeran en los subapartados siguientes.

6.2.1.4. Base de Creación.

Para la deducción de características permanentes del objeto a partir de los atributos de tipo primario. Esta base se aplica una única vez en tiempo de creación de la instancia de TBS y, típicamente, se formulará mediante reglas de inferencia con el formato siguiente para cada propiedad deducida:

$$AP_i, AP_j, ..., AP_n ==> CP_r$$

siendo AP_k el atributo permanente k y CP_r la característica deducida permanente r.

Cabe plantearse el tener formatos preestablecidos como el anterior para cada característica permanente y clase de TBS en un dominio de aplicación, de manera que la persona experta, al crear una TBS particular, como instancia del concepto general, defina en detalle los dominios de las variables y produzca instancias de dichos formatos de la forma:

$$AP_i.\{v_i\}, AP_j.\{v_j\}, ..., AP_n.\{v_n\} ==> CP_r.\{v_r\}$$

siendo:

$AP_k.\{vi\}$: un valor, o conjunto de valores en disyunción, del dominio del atributo primario AP_k.

$CP_r.\{v_r\}$: un valor (o conjunto de valores en disyunción) , del dominio de la característica permanente deducida CP_r.

6.2.1.5. Base de Iniciación.

Para la deducción de valores iniciales de las variables de estado del objeto. Esta base se aplica una vez por cada ejecución de la tarea; es decir, una vez por cada proceso global de simulación de un objeto. En general, se formulará mediante reglas de inferencia con la siguiente estructura inferencial por cada variable de estado:

$$AP_i, AP_j, ..., AP_n, CP_r, CP_s, ..., CP_t, A(t\text{-}m\Delta t, t_0) ==> VE_k(t_0)$$

siendo:

AP_u : atributos primarios.

CP_w : características permanentes deducidas.

$A(t\text{-}m\Delta t, t_0)$: acciones externas durante el intervalo $[t\text{-}m\Delta t, t]$.

$VE_k(t_0)$: variables de estado en el instante inicial (instante actual).

De forma similar a la Base de Creación, el usuario podrá formular reglas de deducción de estados iniciales para cada TBS particular produciendo instancias concretas del formato anterior.

6.2.1.6. Base de Proceso.

Para la modelización de una transición de estado; es decir, para la deducción de valores de las variables de estado en un horizonte temporal igual a un Δt. Ello corresponde a un nivel del árbol de transición de estados presentado en la figura 6j.

Por consiguiente, esta base será utilizada por la TBS para realizar un paso de simulación. Como ya se mencionó anteriormente, el paso de simulación se producirá, en general, de forma no determinista de manera que, a partir de un estado actual (o del estado inicial si se trata del primer paso de simulación) es posible transitar a varios estados siguientes.

La base de proceso es la que contiene, por tanto, el conocimiento sobre el comportamiento dinámico del objeto. Su formulación dependerá del tipo de modelo de que se disponga. Si las transiciones de estado se modelizan mediante transiciones explícitas (es decir, se tiene un modelo superficial) la base se formulará mediante reglas de inferencia que deducirán, a partir de las acciones externas y el estado actual, los nuevos estados con un etiquetado temporal.

Si, por el contrario, se tiene un modelo profundo del comportamiento del objeto, la base se formulará mediante sistemas de restricciones que recojan la forma de evolución del objeto en el tiempo, de acuerdo con el conocimiento general y específico descrito en [Kuipers 86], [DeKleer, Brown 84a] y [Salmerón 92]. La representación, en este caso, se completaría con una base de reglas para la deducción del nuevo estado, a partir del estado actual, el conjunto de acciones externas y la evolución de signos previsible obtenida de la aplicación del sistema de restricciones.

Evidentemente, esta aproximación no es única. Por ejemplo, en [Salmerón 92] se plantean unidades cognitivas basadas en la TBS para la predicción de comportamiento de componentes en una instalación industrial. En esta aproximación, el conocimiento general sobre las evoluciones de las variables, tanto en signo como en magnitud, se representan mediante reglas de inferencia, en tanto que el conocimiento específico que permite filtrar las transiciones se representa mediante un sistema de restricciones cualitativas, totalmente en linea con la propuesta de [Kuipers 86].

De las diferentes formas de representación que pueden ser necesarias en la base de proceso, se deduce la necesidad de que ésta pueda descomponerse en diferentes tipos de subbases, cada una de las cuales:

1) Recoge un esquema de representación y una estrategia.
2) Se comunica con el resto de las subbases, con independencia de su propio formalismo de representación.
3) Coopera con el resto de las subbases para resolver la tarea de razonamiento sobre el objeto.

Este esquema está inspirado en el concepto de "Knowledge Source" (KS), de las arquitecturas de pizarra [Erman et.al. 80], en el que diferentes módulos de conocimiento cooperan para resolver un problema global y comparten información mediante una base de datos general (la pizarra), dentro de una estrategia que puede ser oportunista en mayor o menor medida.

En el caso de la TBS, cada subbase actúa como fuente independiente de conocimiento, con su propio esquema de representación y su propia estrategia, anotando sus resultados en el marco de la tarea, que actúa como pizarra, uniformizando el lenguaje de comunicación.

Nótese como la TBS no impone ninguna limitación al tipo o número de subbases que pueden existir, así como tampoco en lo que respecta al control.

Cada subbase puede producir sus resultados de forma no determinista. Por tanto, debe existir algún mecanismo capaz de evaluar y, eventualmente, rechazar hipótesis intermedias anotadas en el marco de la tarea. La TBS permite representar explícitamente tanto el conocimiento estratégico de selección de subbases como el conocimiento de control necesario para podar resultados intermedios y finales. Este aspecto se recoge mediante la posibilidad de definir subbases de control que gobiernen la ejecución de lo que, globalmente, se entiende como la base de proceso.

Por ejemplo, en [Salmerón 92] se plantea un esquema de representación del comportamiento de componentes de una instalación industrial vía unidades cognitivas, denominadas Unidades de Conocimiento Físico (UCFs), cuya definición está basada en la TBS. En dichas unidades, la base de proceso se estructura según una colección de subbases, cada una de las cuales modeliza la dinámica del componente en un estado de funcionamiento. Una subbase de control selecciona la subbase adecuada de acuerdo con el estado en el que se encuentra el componente. Este esquema de descomposición del componente en regiones de funcionamiento se basa en la técnica de [DeKleer, Brown 84b] y se muestra en la figura 6k.(1)

(1) Figura extraída de [Salmerón 92].

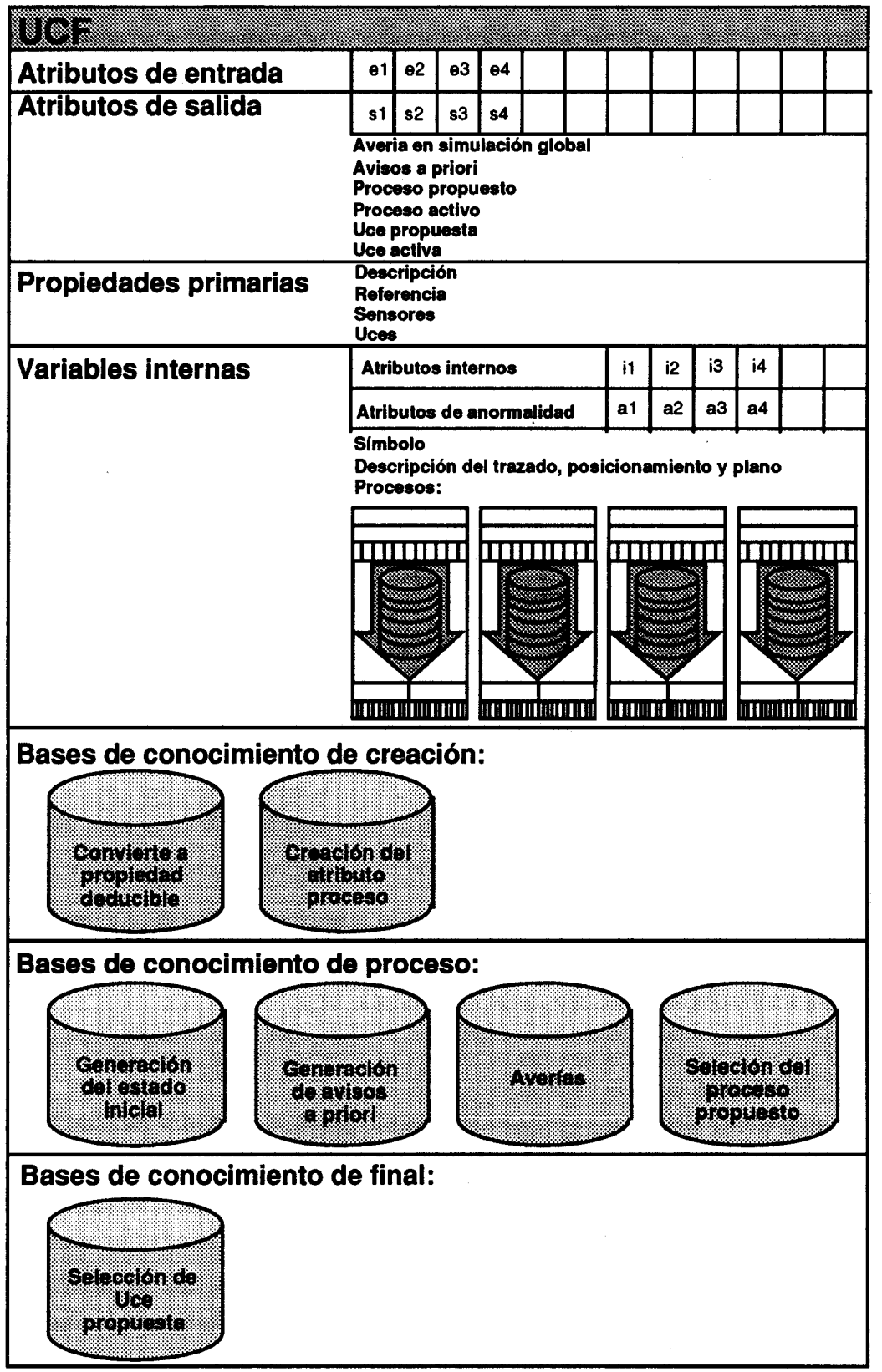


Figura 6k: Unidad de Conocimiento Físico [Salmerón 92] definida sobre una TBS.

La estrategia de selección de subbases podría definirse, incluso, de forma totalmente oportunista. En dominios reales muy complejos, sin embargo, rara vez será esta la estrategia adecuada. La arquitectura no limita esa posibilidad, pero también permite, en el otro extremo, definir estrategias completamente "ad-hoc".

6.2.1.7. Base de Finalización.

Para la deducción de los atributos de salida de la tarea, a partir de los atributos de caracterización del estado. Esta base se aplica una vez por cada ejecución de la tarea. En general, se formulará mediante reglas de inferencia del tipo:

$$VE_1(t), VE_2(t), ..., VE_n(t) \Longrightarrow S_i(t)$$

siendo:

$VE_j(t)$: variable de estado en el instante t.

$S_i(t)$: variable de salida en el instante t.

6.2.2. Estructura de la tarea.

La figura 61 muestra la estructura de tarea para la TBS. El objetivo de la inferencia es generar predicciones en un horizonte temporal determinado y en base a iteraciones de una tarea elemental que infiere, para un periodo básico, los posibles nuevos valores de las variables de estado. El proceso inferencial se estructura en cuatro fases:

1) Fase de creación.

   Esta fase se realiza una vez en tiempo de creación de una instancia de TBS. A partir de los atributos primarios de caracterización física de la tarea, y apoyandose en la Base de Creación, se deducen las características permanentes para su uso en simulación.

2) Fase de iniciación.

   Esta fase se realiza una vez por cada ejecución global de la tarea. A partir de las características permanentes deducidas en fase de creación y de la historia reciente (evolución de las acciones externas al objeto en un determinado horizonte temporal pasado) se deducen, apoyandose en la Base de Iniciación, valores iniciales para las variables de estado, configurando el estado inicial. En general, aquí puede obtenerse un conjunto de posibles estados iniciales dado que, si bien la historia reciente de las acciones externas puede conocerse con precisión, las características permanentes pueden haberse deducido de forma no determinista.

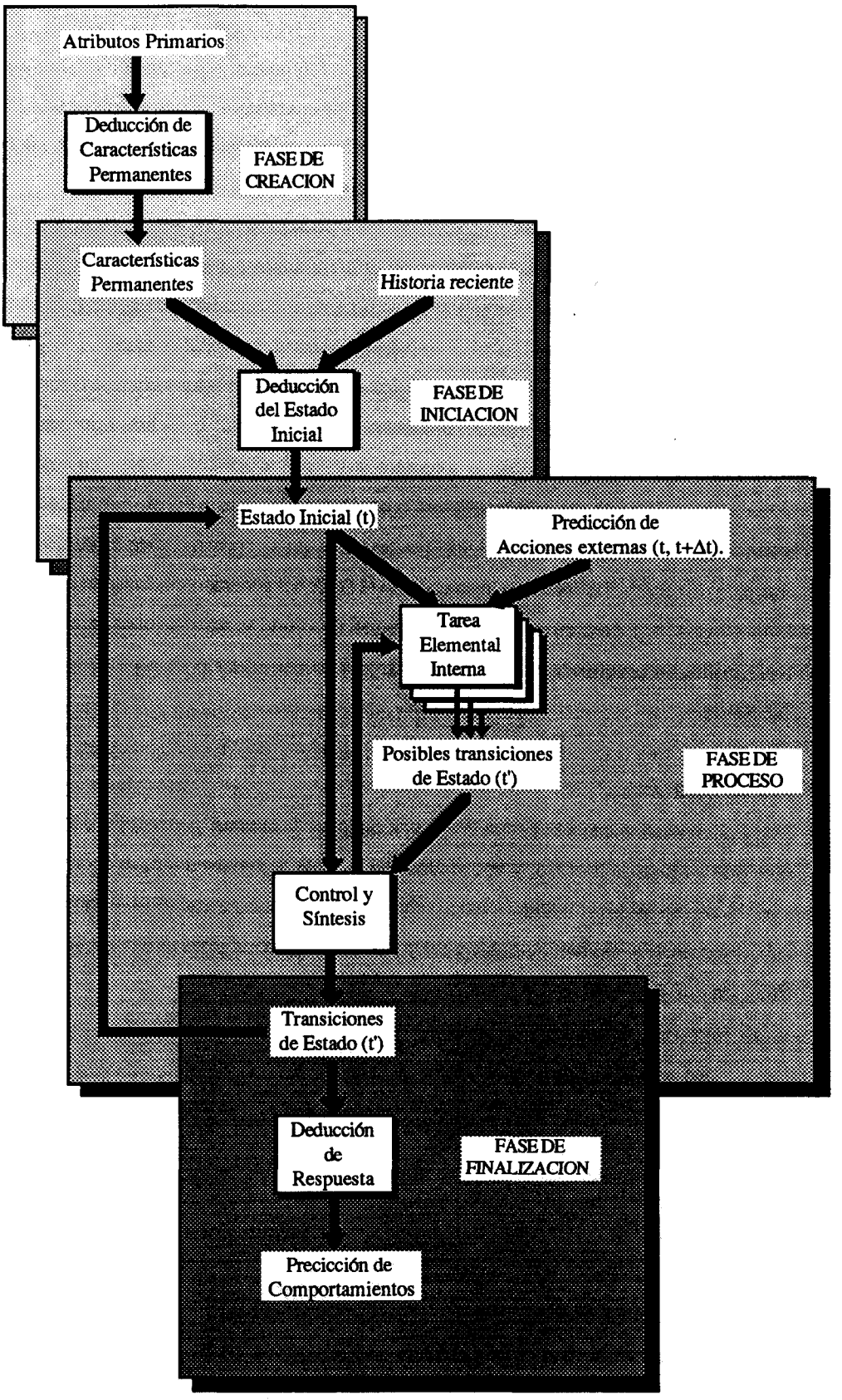


Figura 61: Estructura general de una Tarea Básica de Simulación.

3) Fase de proceso.

Esta fase es la que incorpora el procedimiento de inferencia por simulación cualitativa del objeto físico. La estrategia general itera una tarea elemental interna que, a partir del estado actual (estado inicial en la primera iteración) y la predicción de acciones externas al objeto durante un periodo básico, deduce posibles transiciones de estado para ese periodo.

Para diferentes estados de partida pueden existir diferentes tareas elementales que simulen los posibles comportamientos del objeto. Debido a esto, es necesario que exista una Base de Conocimiento de Control que, a partir del estado en curso, seleccione la tarea elemental adecuada.

Dado que tanto la entrada -predicción de acciones externas- como los parámetros físicos, y por tanto el estado inicial, se definen de forma aproximada, puede obtenerse un árbol de posibles evoluciones futuras cuyos niveles corresponderán a conjuntos de estados alternativos separados en el tiempo por un periodo básico. Para restringir la expansión combinatoria de este árbol se introduce, como parte del control de la tarea, un paso de síntesis que, apoyado en su correspondiente base, retenga un número prefijado de opciones de predicción en cada nivel del árbol generado, rechazando el resto. De forma más general, puede pensarse en un sistema de satisfacción de restricciones que maneje criterios de verosimilitud y/o pertinencia de forma análoga a lo considerado para las Tareas de Generación de Escenarios.

4) Fase de finalización.

Esta fase se realiza una vez por cada ejecución global de la tarea. A partir del árbol de "envisionment" obtenido en fase de proceso, y apoyandose en la Base de Finalización, se deducen los valores de las variables de salida del objeto físico. Puede ocurrir que las mismas variables de estado (o un subconjunto de ellas) actúen como acciones externas a otros objetos descendientes en el sistema de flujo. En este caso esta fase será vacía.

6.3. SISTEMAS BASICOS DE REPRESENTACION E INFERENCIA.

6.3.1. Sistema de Administración de Marcos y Reglas (SAR).

El Sistema de Administración de Reglas y Marcos (SAR) proporciona un entorno para diseño de bases de conocimiento basadas en reglas de inferencia para desarrollar sistemas expertos de primera generación, ofreciendo un eficiente uso de los recursos del ordenador. Su filosofía de actuación asume el papel de componente o subsistema de representación integrado en arquitecturas complejas que utilicen distintos modelos de representación del conocimiento y que, como uno de los componentes unitarios de razonamiento, se aproveche la estructura deductiva del SAR.

SAR ha sido desarrollado, con anterioridad a esta tesis, en el Departamento de Inteligencia Artificial de la Facultad de Informática de Madrid y puede encontrarse una exposición detallada del mismo en [Molina 90]. Lo que sigue es un resumen (breve, por razones de espacio) de dicha exposición.

El SAR aporta un entorno de modelización para formular conocimiento simbólico a través de una estructura clasificativa de los conceptos que componen el dominio de discurso y permite declarar conjuntos de relaciones causa-efecto, expresadas con reglas de inferencia, que posibilitan el razonamiento sobre el dominio. El conjunto conforma lo que se denomina base de conocimiento del sistema, que se caracteriza por:

1) Constituir información externa al sistema lo que le da flexibilidad de modificación y mantenimiento, para su continua puesta a punto y mejora.
2) Permitir la formulación de conocimiento incompleto e incierto, que se presenta en ocasiones en la descripción de problemas, y que se revela de utilidad para diversas aplicaciones.

La descripción del sistema SAR que se realiza en este apartado tiene como objetivo mostrar la capacidad de representación del conocimiento que tiene el sistema, así como los métodos utilizados para llevar a cabo la tarea de razonamiento. El anejo B incluye, como parte del lenguaje DECON, la gramática de definición de reglas y marcos específica del SAR.

6.3.1.1. Representación del conocimiento.

El modelo cognitivo del SAR se basa fundamentalmente en la terna proposicional objeto-atributo-valor, con la que el universo de discurso se estructura según un conjunto de objetos. Entre ellos pueden establecerse relaciones de clase e instancia para representar una jerarquía clasificativa sobre la que definir a continuación un conjunto de reglas de inferencia que describan el comportamiento deductivo de dichos objetos.

Se describen a continuación los componentes del esquema de representación del conocimiento.

La terna objeto-atributo-valor es una unidad de representación clásica en Inteligencia Artificial que modeliza conceptos de forma elemental. Su uso es adecuado para establecer el dominio de razonamiento del sistema, en base a la enumeración y caracterización de los conceptos que lo componen. El objeto es la unidad básica de representación. Los objetos pueden ser físicos (tangibles) o entes conceptuales como actos,categorías abstractas, etc. Se compone de un nombre identificador y de una colección de atributos. Ejemplos de objetos son: persona, préstamo bancario, cuenca hidrográfica, población y tramo de carretera, etc.

El atributo representa una propiedad de un objeto. En el SAR los atributos pueden ser de tres tipos:

1) Discreto; toma valores de un dominio finito definido previamente.
2) Numérico; toma valores enteros o reales.
3) Intervalo; toma como valor un par de números, de forma que el primero sea inferior o igual al segundo.

El concepto de terna objeto-atributo-valor descrito se aproxima al de una proposición lógica, de tal forma que se puede asociar a cada una los significados lógicos VERDADERO o FALSO. En el SAR, se extiende la semántica asociada a una proposición de modo que no se le asigna VERDADERO o FALSO sino un coeficiente de certeza numérico según el modelo MYCIN [Shortliffe 76]. Como es sabido, dicho modelo establece un intervalo continuo de la recta real [-1,+1] para representar la credibilidad que se asigna a una proposición.

El concepto de Marco [Minsky 75] se introduce como propuesta de representación de las situaciones y conceptos estereotipados que habitualmente se presentan en el razonamiento humano. Un marco es una descripción de un objeto con un conjunto de ranuras (los atributos); cada ranura contiene tanto el dominio de valores que puede tomar, como información de cómo obtener su valor, por ejemplo, un conjunto de reglas de inferencia, un valor por defecto, un acceso a base de datos, o incluso un procedimiento "ad-hoc".

La visión del objeto como marco lo eleva de nivel, pasando de ser una estructura pasiva de información a un módulo conceptual dotado de capacidad de razonamiento propia; contiene tanto la información que lo caracteriza como la forma de obtenerla. Esta característica permite la modularización del conocimiento gobernada por los atributos de los objetos, para definir parcelas de razonamiento de distinto origen que conviven en la estructura del objeto. Los métodos de cálculo de atributos disponibles en el SAR son:

1) Preguntado: la evaluación del atributo se realiza preguntando al usuario por consola.
2) Constante: el valor del atributo es invariable a lo largo de las distintas inferencias;

3) Reglas: para deducir el valor del atributo se estudian las reglas que concluyen sobre él.
4) Fichero: el valor se deduce de la búsqueda en fichero.

El conjunto de métodos es abierto; es decir, el usuario lo puede extender cómodamente programando otras funciones.

Los objetos descritos son útiles para definir el universo de discurso sobre el que posteriormente diseñar un esquema de razonamiento según un conjunto de reglas. Sin embargo, es conveniente disponer de formas de agrupación en familias o conjuntos de objetos que tienen información común. Para ello se manejan las clases y las instancias. Una clase es un objeto que define las características de una familia o conjunto de objetos. Cualquier objeto que pertenezca a dicha familia se definirá como instancia de la clase. Todas las características definidas para la clase serán *heredadas* por cada una de las instancias. Además, se pueden definir subfamilias, que son especificación de clase; dichos objetos se llaman subclases y heredan también las características de la clase superior. De esta forma, el marco conceptual se define según una jerarquía de clases, subclases e instancias.

Los descendientes (instancias o subclases) de una clase pueden redefinir particularidades de su superior y suponiendo, por tanto, excepciones a la generalidad.

El razonamiento deductivo se representa mediante reglas de inferencia. Una regla es una relación causa-efecto entre atributos. La regla consta de:

- Antecedente
- Consecuente
- Grado de implicación

El antecedente es una conjunción de premisas de tipo proposicional, es decir, ternas objeto-atributo-valor. El consecuente es una terna objeto-atributo-valor. El grado de implicación es la certeza MYCIN que debe tener la conclusión, en caso de que las premisas tengan certeza total.

Una regla, pues, tiene la forma:

SI objeto-atributo-valor **y**
objeto-atributo-valor **y**
.....
objeto-atributo-valor
ENTONCES [Grado implicación]

objeto-atributo-valor

Las reglas se interpretan de forma que, si se verifican simultáneamente las premisas, puede afirmarse también la conclusión, con el grado de certeza correspondiente.

En el esquema de objetos de SAR, el mecanismo de herencia de las clases permite que los atributos de una instancia hereden los métodos de cálculo de los atributos de niveles superiores. Para el caso particular de evaluación por reglas, el valor de un atributo de una instancia se puede deducir del conjunto de reglas asociado a dicho atributo, bien porque el conjunto de reglas se presenta a nivel de la instancia, o bien porque se herede de una clase superior. Esta posibilidad permite definir conocimiento expresado en reglas al nivel de clase y definir subclases e instancias que comparten dicho conocimiento.

La base de conocimiento deja de representarse, por tanto, como una colección secuencial de reglas y se formula como un conjunto de subbases estructuradas. Esa característica permite el diseño de grandes bases de conocimiento aplicando técnicas de modularización.

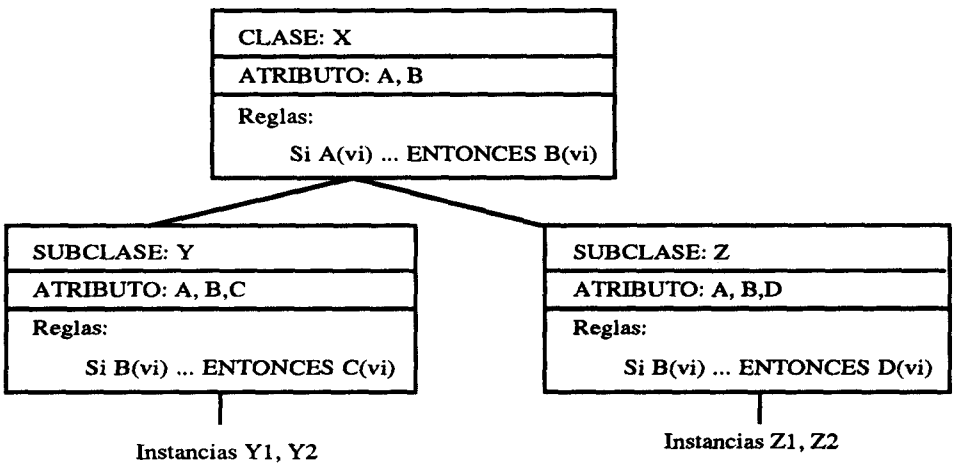


Figura 6m: Esquema de herencia del SAR.

La figura 6m representa el esquema de herencia de reglas del SAR. Cualquier instancia de Y que se interese por el valor del atributo C provoca el cálculo de dicho atributo heredando el conjunto de reglas de la subclase Y, y como consecuencia del cálculo de B, se hereda asimismo el conjunto de reglas de la clase X. Del mismo modo, una instancia de Z de la que se pretenda averiguar el valor de su atributo D, hereda el conjunto de reglas que deducen D, de la subclase Z, así como el conjunto de reglas que deducen B, de la clase X.

El ejemplo anterior pone de manifiesto el uso de un conocimiento común desde distintos ángulos evitando así replicar conjuntos de reglas en cada una de las instancias. Sin embargo es posible el tratamiento de excepciones,

es decir, instancias que se comportan de forma distinta a la clase. En esos casos la presencia de un conjunto de reglas a nivel de las instancias o subclases marca el comportamiento las mismas, sin ascender al conjunto de reglas de la clase general.

6.3.1.2. Estrategia de razonamiento.

Según se ha descrito en el apartado anterior la base de conocimiento del SAR se construye mediante marcos, organizados según una jerarquía de clases e instancias, y una base de reglas que define el comportamiento deductivo.

En este apartado se describe el método de navegación por la estructura de la base de conocimiento y las operaciones que se realizan para acumular evidencia a lo largo del recorrido. Posteriormente se muestran los modelos de transmisión de evidencia.

El motor de inferencia es el procedimiento automático que navega por la base de conocimiento definida explícitamente. El motor es de tipo "backward" o regresivo, de manera que se parte de los objetivos que se pretenden deducir y a través de las reglas se encadena hacia los datos. El proceso tiene como punto de comienzo la evaluación de un atributo. Para ello dispara su método de cálculo asociado, que toma control para deducir su(s) valor(es). Tiene, además, como cometido importante, la transmisión automática de conocimiento a través de la jerarquía de clases e instancias conforme al mecanismo de herencia.

El método mas relevante es el de evaluación y disparo de reglas. Este método recorre las reglas en sentido regresivo, analizando aquéllas que concluyen sobre el objetivo, para disparar las que superen cierto umbral de certidumbre. El estudio de dicho conjunto de reglas involucra, en general, la evaluación de otros atributos que, planteados esta vez como subobjetivos, encaminan al motor de inferencia hacia los datos. Una vez alcanzados los atributos dato, se realiza adecuadamente la transmisión de evidencia para concluir el valor o conjunto de valores deducidos con su certeza asociada, para el atributo de partida. El procedimiento que dirige este proceso tiene las siguientes etapas:

1) Sea R el conjunto de reglas que concluyen sobre el objetivo. Este conjunto puede ser el que se haya definido para el objeto que se pretende consultar, o en su defecto, un conjunto de reglas heredado de alguna clase de niveles superiores.
2) Para cada regla del conjunto R se evalúa cada una de las premisas instanciándolas para el objeto sobre el que se realice la consulta. Si alguna resulta con certeza inferior a un cierto umbral se detiene la evaluación de premisas. Sea CF la mínima certeza de todas las premisas.
3) Si CF supera el filtro 0.2, disparar la regla:

- Acumular certeza sobre la conclusión.
- Anotar la regla como disparada.

4) Finalmente se realiza un proceso de normalización que distribuye la certidumbre adecuadamente en los valores excluyentes de los atributos.

Como ya se ha apuntado, el tratamiento de la incertidumbre en el SAR se realiza mediante el modelo MYCIN. Su formulación se resume en la tabla adjunta.

Para definir un factor de certidumbre se parte de las probabilidades subjetivas asociadas a los hechos P(h), y de probabilidades condicionadas de las reglas; es decir, a la regla "si e entonces h" se asocia P(h/e). Se definen la me de credibilidad de la hipótesis h dada la evidencia e (MC), y la medida de incredibilidad de h dada e (MI) de la forma

$$MC = \frac{\max(p(h,e), p(h)) - p(h)}{1 - p(h)} \quad \text{si } p(h) <> 1$$

$$MC = 1 \quad \text{si } p(h) = 1$$

$$MI = \frac{p(h) - \min(p(h,e), p(h))}{p(h)} \quad \text{si } p(h) <> 1$$

$$MI = 1 \quad \text{si } p(h) = 1$$

Con estas dos medidas se define el factor de certeza CF = MC - MI, que toma valores entre -1 (totalmente falso) y · (totalmente cierto). Dadas dos reglas:

Si e1 entonces h (CF1)
Si e2 entonces h (CF2)

Suponiendo independencia entre premisas, el CF acumulado se obtiene:

Si CF1 y CF2 >= 0 CF= CF1 + CF2 - CF1*CF2
Si CF1 y CF2 <= 0 CF= CF1 + CF2 + CF1*CF2
Si CF1 y CF2 tienen signos opuestos:

$$CF = \frac{CF1 + CF2}{1 - \min(|CF1|, |CF2|)}$$

El factor de certeza de los conectivos es:

CF (A y B) = min (CF(A), CF(B))
CF (A o B) = max (CF(A), CF(B))
CF (no A) = max [CF(X) / X≠ A] = CF(X1vX2v...vXm)/Xk ≠ A

El tratamiento de la incertidumbre se ha completado utilizando la teoría de la Evidencia de Dempster-Shafer, que modeliza la incertidumbre en un conjunto finito de hipótesis mutuamente excluyentes y exhaustivo. Este paradigma parece adecuado para el reparto de la certeza aportada por las reglas MYCIN en los atributos con valores de tipo excluyente [Molina 90]; Especialmente, cuando se tiene evidencia en contra de una hipótesis, el método asigna convenientemente dicha certeza entre el resto de alternativas. El método que se aplica es la formulación aportada por Barnett, que simplifica los cálculos bajo las condiciones que se dan en el modelo de representación utilizado en el sistema SAR. En la tabla adjunta se presenta un resumen.

Dado un atributo con n valores, vi (i=1,..,n) se calculan las medidas:

Si = 1 - (1 - s1)(1 - s2)...(1-sk)
S'i = 1 - (1 - t1)(1 - t2)...(1-tm)

donde sj=gj*Cr, siendo gj el grado de implicación de la regla que concluye sobre vi con gj>0 (a favor) y Cr credibilidad de las premisas de dicha regla, y tj=gj*Cr, siendo gj el grado de implicación de la regla que concluye sob vi con gj<0 (en contra) y Cr la credibilidad de sus premisas. Además se obtienen:

$$p_i = \frac{S_i (1 - S'_i)}{1 - S_i \cdot S'_i} \qquad c_i = \frac{S'_i (1 - S_i)}{1 - S_i \cdot S'_i}$$

ri = 1 - pi -ci
di = ci + ri

y a continuación se aplican las fórmulas:

$$1/k = \prod_i d_i \left(1 + \sum_i \frac{p_i}{d_i}\right) - \prod_i c_i \qquad d_i \neq 0$$

$$Cr(v_i) = k \left(p_i \prod_{j \neq i} d_j + r_i \prod_{j \neq i} c_j\right)$$

La normalización se realiza al final de la evaluación del conjunto de reglas que concluyen sobre un atributo y siem que los valores de su dominio sean de tipo excluyente.

6.3.2. Sistema de Satisfacción de Restricciones (SSR).

El sistema de satisfacción de restricciones (SSR) constituye un entorno para la definición y evaluación de sistemas de restricciones de tipo cualitativo. Se ha concebido, de manera análoga al sistema SAR comentado en el apartado 6.3.1, como pieza integrable en esquemas de representación más complejos (en este caso como módulo básico de representación dentro de arquitecturas de segunda generación).

El SSR ha sido desarrollado, en paralelo a esta tesis, en el Departamento de Inteligencia Artificial de la Facultad de Informática de Madrid, y se puede encontrar su exposición en detalle en [Izquierdo 92]. Lo que sigue es un breve resumen de dicha exposición.

El SSR consta básicamente de:

1) Un lenguaje de especificación de restricciones.
2) Un motor capaz de interpretar esta especificación y de evaluar sus respuestas.

En los subapartados siguientes se comenta la representación del conocimiento y la estrategia de razonamiento. La gramática del lenguaje de especificación se incluye en el anejo B como parte del lenguaje DECON.

6.3.2.1. Representación del conocimiento.

En un sistema de satisfacción de restricciones la programación de una solución es una tarea declarativa: el programador establece relaciones entre un conjunto de objetos, siendo responsabilidad del sistema encontrar las soluciones que satisfacen estas relaciones. Por tanto la labor de modelización consiste en la identificación de las variables relevantes y la definición de las relaciones existentes entre ellas. En el caso particular de restricciones cualitativas las variables tomarán valores de entre una gama discreta o espacio cualitativo en el sentido de [DeKleer, Brown 84]. Por tanto, un espacio cualitativo es un dominio discreto ordenado, que supone una discretización del dominio real a unos pocos valores representativos o valores hito. En el caso de parámetros numéricos, los espacios cualitativos definen una partición de la recta real, de tal forma que las variables tomarán valores significativos desde el punto de vista de la aplicación.

Entre los valores de un espacio cualitativo se asume por defecto una relación de orden parcial establecido en función de la posición relativa en que se declaren. Definido, por ejemplo, el espacio DERIVADAS como el conjunto de valores {-, 0, +} se asume que:

- "menor" 0 "menor" +
+ "mayor" 0 "mayor" -

y dado que los valores son de tipo cualitativo, no hay criterio para decidir cual es la relación de orden entre dos variables con el mismo valor cualitativo. Así por ejemplo, todos los siguientes casos pueden ocurrir:

- "menor" -
- "mayor" -
- "igual" -

En el SSR, los operadores relacionales utilizados en la expresión de una restricción (<, <=, =, >=, >, <>) se refieren siempre a este orden preestablecido, no obstante el usuario tiene capacidad para modificar la semántica de estos operadores si dispone de conocimiento específico adicional. Por ejemplo, en determinados casos particulares podrá asumirse que + es (exactamente) igual a +.

Los espacios cualitativos admiten la definición de operaciones binarias cerradas. Una operación se define como una aplicación del producto cartesiano del dominio sobre el conjunto de partes del mismo (DxD→ P(D)). Esta definición no tiene porque ser completa (la operación puede no tener sentido para un par de valores concretos) ni determinista (el resultado de una operación puede ser un conjunto de valores posibles). Por ejemplo, en el espacio de las Temperaturas, podría definirse la suma de la forma siguiente:

OPERACION: +

ASOCIADA A: Espacio cualitativo **Temperaturas**

TIPO: Conmutativa

<muy fría> + <muy fría> = <muy fría>; <muy fría> + <fría> = <muy fría>;

<muy fría> + <normal> = <muy fría> <fría>: etc.

La definición completa se detalla en la tabla adjunta.

+	**muy fría**	**fría**	**normal**	**caliente**	**muy caliente**
muy fría	muy fría	muy fría	muy fría fría	fría normal	fría normal caliente
fría	muy fría	muy fría	fría	normal	normal caliente
normal	muy fría fría	fría	normal	caliente	muy caliente
caliente	fría normal	normal	caliente	muy caliente	muy caliente
muy caliente	fría normal caliente	normal caliente	muy caliente	muy caliente	muy caliente

Las restricciones establecen, en general, las relaciones entre las variables cualitativas del dominio. En el caso particular de la simulación cualitativa de sistemas físicos, las restricciones expresan el comportamiento dinámico. De hecho, la adquisición del conocimiento para este caso consiste básicamente en relajar las ecuaciones diferenciales utilizadas en la física clásica.

Las restricciones, en el SSR, se formulan en términos de operaciones y variables cualitativas. Los operadores relacionales utilizables en la ecuación de una restricción son: <, <=, =, >=, >, <>. La semántica de estos operadores estará siempre referida al orden asumido entre los valores de un dominio . Por razones de eficiencia y simplicidad en el diseño e implementación de los algoritmos de satisfacción de restricciones, el sistema sólo admite restricciones de tipo binario, en cualquiera de las formas siguientes:

1) X § Y <-> Z, siendo X, Y y Z variables o constantes definidas sobre el espacio cualitativo, § una operación definida sobre el mismo espacio y <-> un operador relacional.
2) X <-> Y, siendo X e Y dos variables o constantes y <-> un operador relacional.

El hecho de contar solo con restricciones de tipo binario no supone pérdida de generalidad: restricciones que involucren N variables pueden reducirse a N-2 restricciones del tipo 1), utilizando variables auxiliares.

La conversión de restricciones arbitrarias a restricciones binarias se realiza de forma automática como paso previo al proceso de satisfacción, permitiendo que el usuario se centre en el dominio y en como modelizarlo sin hacer consideraciones de implementación.

La modelización de un problema en términos de restricciones queda, pues, determinada por los siguientes componentes:

1) Un **espacio cualitativo** que, a su vez, consta de:
   - un conjunto de **valores cualitativos**.
   - un conjunto de **operaciones** cerradas sobre el dominio.
   - una relación de orden entre los elementos del dominio.
2) Un conjunto de **restricciones** de tipo binario.

6.3.2.2. Estrategia de razonamiento.

Las restricciones al poner de manifiesto las interdependencias entre las variables del dominio, limita los valores que estas pueden tomar. De esta forma, si el valor de una variable está determinado, es posible afirmar que el resto de parámetros relacionados con él sólo podrán tomar valores cualitativos coherentes con el conjunto de restricciones definidas.

De esta forma la representación en restricciones se muestran como un vehículo de razonamiento simbólico no determinista. La técnica basada en restricciones se puede utilizar de forma integrada con otros modelos de razonamiento en sistemas complejos. Por ejemplo, en razonamiento sobre sistemas físicos, se pueden definir bases de reglas para deducir los dominios iniciales de algunas variables para, a continuación y por medio de restricciones, averiguar todos los posibles estados en que se pueden encontrar el resto de variables. La salida del sistema de restricciones serán las interpretaciones (asignaciones individuales de valores a variables de forma

consistente con el conjunto de las restricciones), cada una de las cuales constituye realmente un posible estado del sistema.

Una diferencia del modelo de razonamiento de los sistemas de restricciones es la ausencia de una direccionalidad clara en la línea de inferencia, en contraposición a otras formas de razonamiento automático como, por ejemplo, los sistemas de producción o los sistemas basados en reglas .

En la literatura aparecen numerosas propuestas de algoritmos de satisfacción de restricciones, todas ellas encaminadas a reducir el espacio de búsqueda durante el proceso de satisfacción. Se pueden destacar los siguientes tipos:

1) Que buscan establecer un cierto grado de consistencia previo al proceso de propagación. Se distinguen las que producen **consistencia de arco** [Mackworth, Freuder 85], [Mohr, Henderson 86], [Winston 84] y las que producen **consistencia de camino** [Mackworth, Freuder 85], [Mohr, Henderson 86] y [Ching-Chih, Chia-Hoang 86].
2) Que buscan transformar el grafo de partida en otro que garantice una búsqueda sin retrocesos, distinguiendose las que se basan en establecer **consistencia direccional** y **consistencia adaptativa** [Dechter, Pearl 88].
3) Algoritmos simultáneos en los que, al mismo tiempo que se persigue reducir el espacio de búsqueda, se van generando las soluciones [Haralick, Elliot 80].

En este último grupo se inspira el algoritmo utilizado en el SSR, si bien adaptado a la representación en forma ecuacional (no necesariamente binaria) vista en el apartado anterior.

El algoritmo se denomina **División en Subproblemas (DS)** [Izquierdo 92] ya que, en cada paso, intenta centrarse sobre un conjunto de subproblemas de menor entidad que uno dado. Bajo esta aproximación, en cada estado del problema se plantean divisiones en subproblemas sucesivamente más pequeños hasta alcanzar un problema trivial. En este punto cada variable tendrá asignado un único valor y por tanto es directamente una interpretación candidata a ser solución del sistema, y lo será finalmente si satisface la restricciones. Esta técnica está en la linea de los algoritmos simultáneos que preven el futuro ya comentados.

El algoritmo utiliza el filtrado de Waltz para asegurar cierto grado de consistencia entre los valores de los dominios de las variables a la vista de las restricciones. En los apartados siguientes se presentan el filtrado de Waltz y el algoritmos DS.

Filtrado de Waltz.

El objetivo del filtrado es realizar una propagación local de las restricciones para obtener un grado de consistencia parcial en el problema. El grado de consistencia alcanzado, denominado arco-consistencia, es tal que cualquier valor del dominio de una variable que no pueda formar parte de ninguna interpretación consistente con las restricciones, se elimina, de forma que el espacio de búsqueda de soluciones se puede reducir, en general, de manera importante. .

El proceso consiste en examinar una a una las restricciones del problema, de tal forma que se eliminen los valores de los dominios de las variables involucrados en la restricción cuando sean inconsistentes. Dada la restricción X § Y = Z, donde § es una operación definida sobre el dominio cualitativo de las variables X, Y, Z, se dirá que el valor x del dominio de X ($x \in D(X)$) es inconsistente cuando no existan $y \in D(Y)$ y $z \in D(Z)$ tal que X § Y = Z. Por lo tanto, el valor x puede ser eliminado puesto que nunca podrá formar parte de una solución del problema (ya que siempre será inconsistente con esta restricción). Se define, en consecuencia, el proceso EXAMINAR (R), donde R es una restricción de la forma X § Y = Z, que purga los valores inconsistentes con R de los dominios de X, Y, Z.

Después de examinar cada restricción R es necesario realizar su **propagación** de forma local. Esto es, hay que volver a reexaminar todas las restricciones que afecten a alguna variable cuyo dominio se haya visto restringido al examinar R. Para una restricción R, este proceso se denomina PROPAGACION-LOCAL(R).

En cualquier caso, durante el desarrollo de este proceso puede ocurrir que el dominio de alguna variable quede vacío, en cuyo caso se acaba concluyendo que no existe etiquetado consistente para las variables.

El filtrado en un sistema de restricciones asegura que cada valor del dominio de una variable que ha sobrevivido al proceso forma parte de al menos una interpretación parcial consistente en el ámbito de cada una de las restricciones que le afectan por separado, pero puede no existir una solución global de conjunto que satisfaga todas las restricciones simultáneamente. Esto es así por las conexiones entre restricciones que afectan a las mismas variables, de tal forma que si bien cada restricción por separado es capaz de etiquetar las variables locales consistentemente, puede no existir una asignación global para el conjunto completo de restricciones. En particular, esta situación podrá aparecer cuando se defina un modelo en restricciones que sea inconsistente de forma intrínseca. Por ejemplo, supóngase el siguiente sistema en restricciones:

$$X = Y$$
$$X \neq Y$$

donde inicialmente D(X) = {1, 2, 3} y D(Y) = {1, 2}. El filtrado del sistema obtendría los siguientes dominios restringidos: D(X) = {1, 2} y D(Y) = {1, 2}. Sin embargo el sistema es intrínsecamente insatisfacible, si bien cada restricción por separado admite un par de interpretaciones parciales.

Algoritmo DS.

Cuando el filtrado de Waltz no concluye que el sistema es insatisfacible, al menos reducirá la complejidad del problema (al eliminar algunos valores en los dominios de las variables). La propuesta de este algoritmo es un proceso en el que en cada paso se plantea la solución de varios subproblemas más sencillos de forma que la suma de interpretaciones de cada uno de ellos sea el conjunto de soluciones del problema inicial.

Piénsese en una representación en árbol donde cada nodo representa un problema a resolver. Este problema se subdivide pivotando sobre una variable concreta, de forma que los subproblemas que se plantean tan solo difieren en el dominio de esta variable, que en cada uno de ellos tomará un elemento disjunto de un cierto particionado sobre el dominio inicial. A continuación cada uno de los subproblemas es filtrado por separado, de tal forma que cada uno puede obtener como resultado:

1) Insatisfacibilidad, en cuyo caso esa rama del árbol no dará lugar a nuevos subproblemas.
2) Un nuevo problema a estudiar de menor envergadura que el inicial puesto que la reducción del dominio de la variable pivote habrá propagado su efecto, a través de las restricciones, al resto de variables.

De esta forma cada nuevo subproblema se descompone de forma recursiva hasta llegar a una de las condiciones de terminación de una rama:

1) Problema vacío. Esta rama no contiene soluciones consistentes puesto que se ha detectado insatisfacibilidad.
2) Problema trivial. Cada variable tiene asignado un dominio unitario, se ha encontrado por tanto una solución del problema inicial.

La suma de todas las soluciones a las que se llegue en las hojas del árbol constituyen realmente todas las interpretaciones consistentes del problema inicial en la raíz del árbol. En la figura 6n se muestra gráficamente la estrategia de solución de este algoritmo.

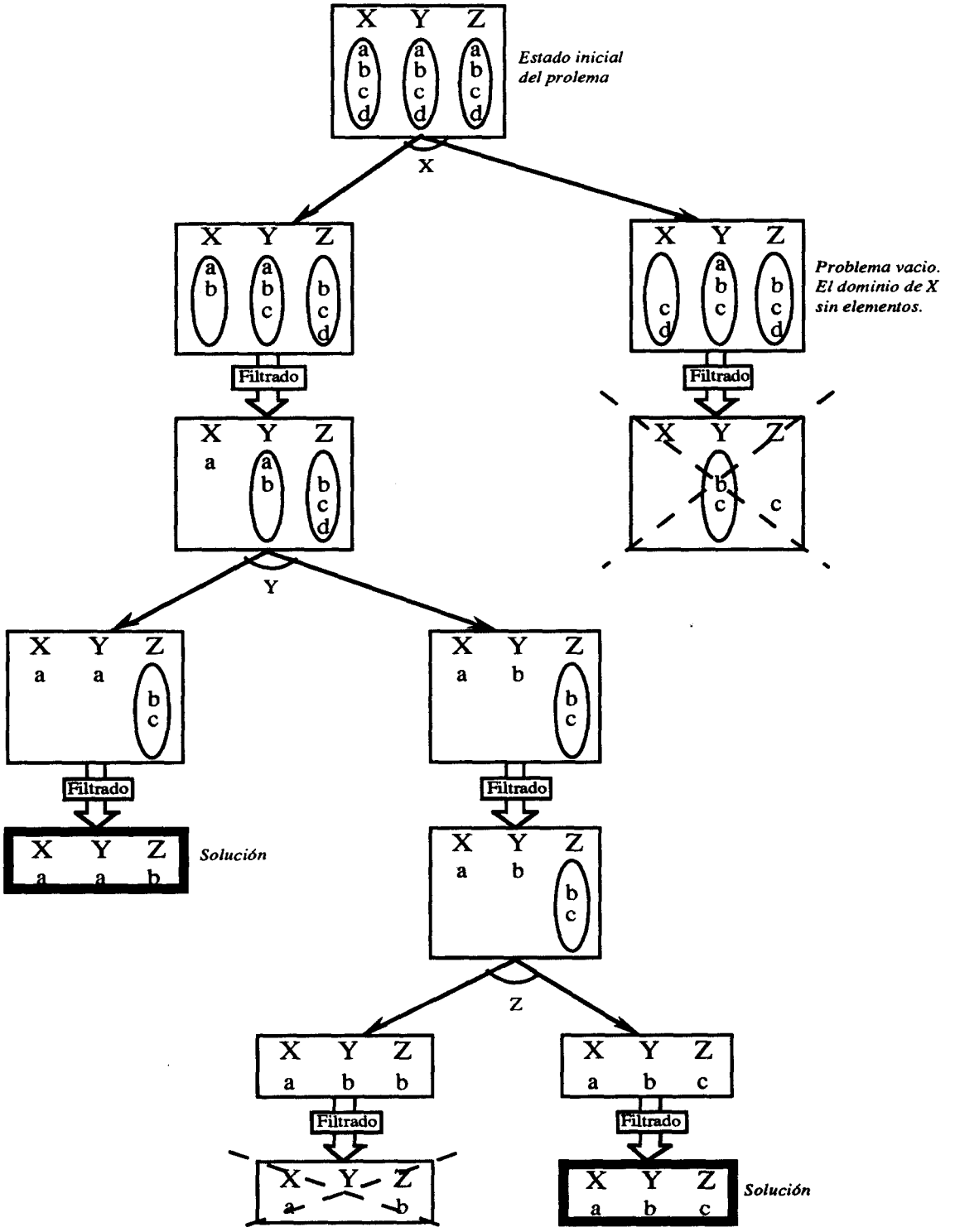


Figura 6n: Satisfacción de Restricciones por División en Subproblemas[1].

1 Figura extraída de [Izquierdo 92]

Dado que las restricciones no varían a lo largo de todo el proceso, lo que caracteriza realmente el problema actual en un nodo son los dominios de cada variable. Esta circunstancia debe ser recordada puesto que el recorrido se plantea con preferencia en profundidad, y por tanto en un nodo genérico, después de investigar una rama, será necesario restituir el estado que caracteriza el problema en el nodo para poder investigar otras ramas.

El algoritmo tal y como se ha planteado cae dentro de la categoría de estrategias de solución simultaneas ya comentadas, es decir de aquella clase de algoritmos que van reduciendo el espacio de búsqueda a la vez que lo van recorriendo para investigar soluciones. El algoritmo se centra en los dominios de las variables, los cuales investiga por grupos de valores, aplicando el filtrado en cada paso.

A continuación se especifica el seudocódigo del algoritmo. La rutina *SUBDIVIDE()* es la que propone una descomposición de un problema dado.

SUBDIVIDE (Problema)
Decidir la variable pivote. (se escoge una variable cuyo dominio actual admita alguna partición, o sea que no sea unitario).
    Particionar el dominio de la variable pivote.
Crear una lista de subproblemas, cada uno de los cuales asigna una de las particiones como dominio de la variable pivote.
    Aplicar a cada subproblema el Filtrado de Waltz.
    Devolver la lista de subproblemas.

Basándose en esta rutina, el proceso recursivo investiga una lista de subproblemas a la vez que va generando una lista de soluciones:

ALGORITMO_DS (Problemas, Soluciones)
Para todo problema P ∈ Problemas:
    Si P es trivial: Añadir P a la lista de soluciones
    Si P es vacío: No se investigan sus soluciones. Pasar al siguiente.
    en otro caso: ALGORITMO_DS (SUBDIVIDE(P), Soluciones)

Como ya se apuntó anteriormente, la filosofía de diseño del SSR le asigna el papel de componente integrable en arquitecturas complejas con otros esquemas de representación, en la linea habitual de trabajos del Departamento de Inteligencia Artificial. De esta forma, el SSR puede utilizarse de forma conjunta con el entorno SAR y, ambos, como sistemas básicos de representación e inferencia en el seno de las Tareas (TBSs, TGEs, TGC, etc) componentes del tipo general de arquitecturas que la tesis propone.

6.4. DEFINICION DEL SOFTWARE.

Aunque la memoria recoge básicamente un trabajo de investigación en Inteligencia Artificial, es preciso afirmar que detrás ha existido una labor de Ingeniería y desarrollo de software casi tan relevante como pueda serlo la misma tesis. La construcción del software que ha servido de soporte a esta investigación ha sido el trabajo de ocho personas a lo largo de cuatro años (periodo 87-91) dentro del Laboratorio de Sistemas Inteligentes.

Como ya se explicó en el capítulo 1 (ver también apéndice A), este proyecto de investigación se encuadra dentro del programa SAIH, con el objetivo de diseñar y construir un Sistema Inteligente de RAzonamiento Hidrológico (SIRAH), de cuya generalización surge el tipo de arquitectura propuesta en la tesis. Respecto del software, el sistema SIRAH tenía como principales requerimientos y restricciones los siguientes:

1) Transportabilidad del software generado. Esto era necesario ya que el entorno a construir debía instalarse, previsiblemente, en varias Cuencas Hidrográficas de España y, en consecuencia, integrarse en diferentes sistemas de información con distintos soportes tanto hardware como software.
2) Eficiencia. El sistema a construir tenía especificaciones de tiempo real.
3) Modularidad y ocultación de información. Estas condiciones de desarrollo eran imprescindibles, si se tiene en cuenta la complejidad y volumen del software que se debía producir, así como a efectos del control de un proyecto realizado por un grupo grande de personas. El hecho de considerar una solución modular conduce inevitablemente a la formulación de muchos niveles de abstracción en el sentido de [Wasserman 83] y, dada la naturaleza del sistema (basado en el conocimiento), ambas características debían ser aplicables tanto a la componente dinámica del software (procedimientos) como a la componente estática (información en general).

   La condición de modularidad aplicada al desarrollo de un sistema complejo conduce al concepto de la ocultación de la información como principio de desarrollo asociado. Ese principio [Parnas 72] establece que los módulos deben diseñarse de forma que la información que manejen sea inaccesible para aquellos otros módulos que no la necesiten.
4) Fiabilidad. Esta condición implicaba, como objetivo, obtener una tasa de errores pequeña tanto en desarrollo como en mantenimiento de programas. Lógicamente, esta condición siempre es deseable en todo desarrollo. Habida cuenta de la poca experiencia de las personas que integraban la fuerza de programación (en su mayoría estudiantes de últimos cursos de Informática) resultaba claro que la fiabilidad debía conseguirse a través del método de trabajo y no de las personas. En teoría, el propio diseño modular y el encapsulamiento de la información debían contribuir al objetivo de fiabilidad.
5) Reutilización del software. Este proyecto era -o debiera haber sido-, ante todo, un proyecto de investigación. Ello implica necesariamente la existencia de una componente importante de desarrollo

por el método de exploración, condicionado a la existencia de diferentes lineas de investigación para el diseño de la arquitectura cognitiva. Inevitablemente, el abandono de una determinada linea de investigación conducía a la eliminación de una componente (a veces importante) del software. Era necesario, por tanto, definir un método de diseño y programación que, ante esta eventualidad:

- Evitara la redefinición continua de módulos ya programados, debiendo ser capaz de traducir, en consecuencia, un cambio en la linea de investigación principal en la simple eliminación de un cierto número de módulos.
- Produjera, en consonancia con la condición anterior, la eliminación de un número relativamente pequeño de módulos.

Las dos primeras condiciones hicieron desistir pronto del uso de entornos y maquinas especializadas de IA, eligiendose finalmente como lenguaje de desarrollo el lenguaje C, por sus características de progresiva estandarización y, aún siendo un lenguaje de bloques, por la facilidad para generar con él código eficiente.

Respecto de los restantes requerimientos, cualquier metodología de diseño y/o programación permite satisfacerlos en teoría, ya sean métodos orientados al proceso o al dato. Finalmente se eligió el método de diseño y programación orientado a objetos, debido principalmente a tres consideraciones:

1) Es el método que procura de forma más natural las características de abstracción, modularidad, encapsulamiento y reutilización exigidas en el software y que redundan en su fiabilidad, tal y como proponen [Wienner, Sincovec 84].
2) Probablemente, es el método más adecuado para implementar conceptos de Inteligencia Artificial derivados de los objetos estructurados para la representación del conocimiento tal y como los describe [Nilsson 80], en particular por superar la tradicional frontera entre los conceptos de dato y proceso, proporcionando un método único para el tratamiento integrado de ambos aspectos del software.
3) Permite una visión unificada del software [Pressman 88]. Este aspecto se ha revelado especialmente ventajoso al producir software para implementación de tareas. Así, las tareas se concibieron primeramente a nivel del conocimiento que debían albergar y del comportamiento que debían exhibir; es decir, como abstracciones funcionales. Desde el punto de vista de la instrumentación, la aproximación por objetos permite producir directamente unidades programables, a partir de dichas abstracciones, con relativamente poco esfuerzo de diseño adicional.

El método de diseño orientado al objeto emerge durante los últimos quince años. De hecho, alguno de los métodos orientados a las estructuras de datos tienen ya elementos de los métodos orientados al objeto. Es en la década de los ochenta, coincidiendo con la rápida evolución de lenguajes como Smalltalk [Goldberg et al. 83] y

Ada [Booch 83] y a partir de los trabajos de [Abbott 83], [Booch 86], [Cox 85, 86] y, más recientemente [Meyer 88], cuando el diseño y programación orientados a objetos cobra interés creciente, hasta el extremo de poderse afirmar que los lenguajes orientados a (o basados en) objetos están actualmente de moda, surgiendo incluso versiones orientadas al objeto de lenguajes clásicos, como es el caso de C++ o Coobol.

Dada esta innegable (y merecida) popularidad, es difícil establecer criticas de este método. Sin embargo, el trabajo desarrollado permite resaltar dos aspectos negativos:

1) Sobre-dimensionamiento del software. En efecto, el hecho de que en el diseño de un objeto no se prejuzgue el uso posterior que de él se hará suele deparar costes de desarrollo superiores. En proyectos de investigación pura (con gran volatilidad del software) este aspecto puede llegar a poner en duda la eficacia del método.
2) Reutilización del software. Si bien esta ventaja se hace más patente bajo este método, en particular si se programan características como polimorfismo y herencia, la afirmación de [Cox 85] en el sentido de que la reutilización de software se alcanzará construyendo un catálogo de "componentes del software" (objetos) en lugar de bibliotecas convencionales, quizá sea optimista en exceso. La experiencia de este proyecto revela que rara vez se reutiliza un componente de software sin mantenimiento adicional. Cuando el diseño es complejo, y el volumen de software posterior grande, este aspecto se vuelve muy significativo, hasta el extremo de que un porcentaje apreciable del personal de desarrollo (del orden del veinticinco por ciento) termina encontrandose permanentemente dedicado al mantenimiento de objetos ya programados.

A pesar de los comentarios anteriores cabe afirmar, a partir de la experiencia adquirida, que la aproximación por objetos (aplicada tanto al diseño como a la programación) mejora de forma significativa la calidad del software producido.

6.4.1. Metodología general de construcción del software.

En la figura 6o se recoge la cronología del desarrollo efectuado. En las sucesivas capas concéntricas se muestran los diferentes niveles para la producción del software, ejecutadas en paralelo, así como los métodos de soporte en cada caso.

Cabe distinguir cuatro estadios del desarrollo:

1) Software básico.
2) Software para implementación de sistemas básicos de representación e inferencia.

3) Software avanzado para implementación de tareas.
4) Software de aplicación e interfaces de usuario.

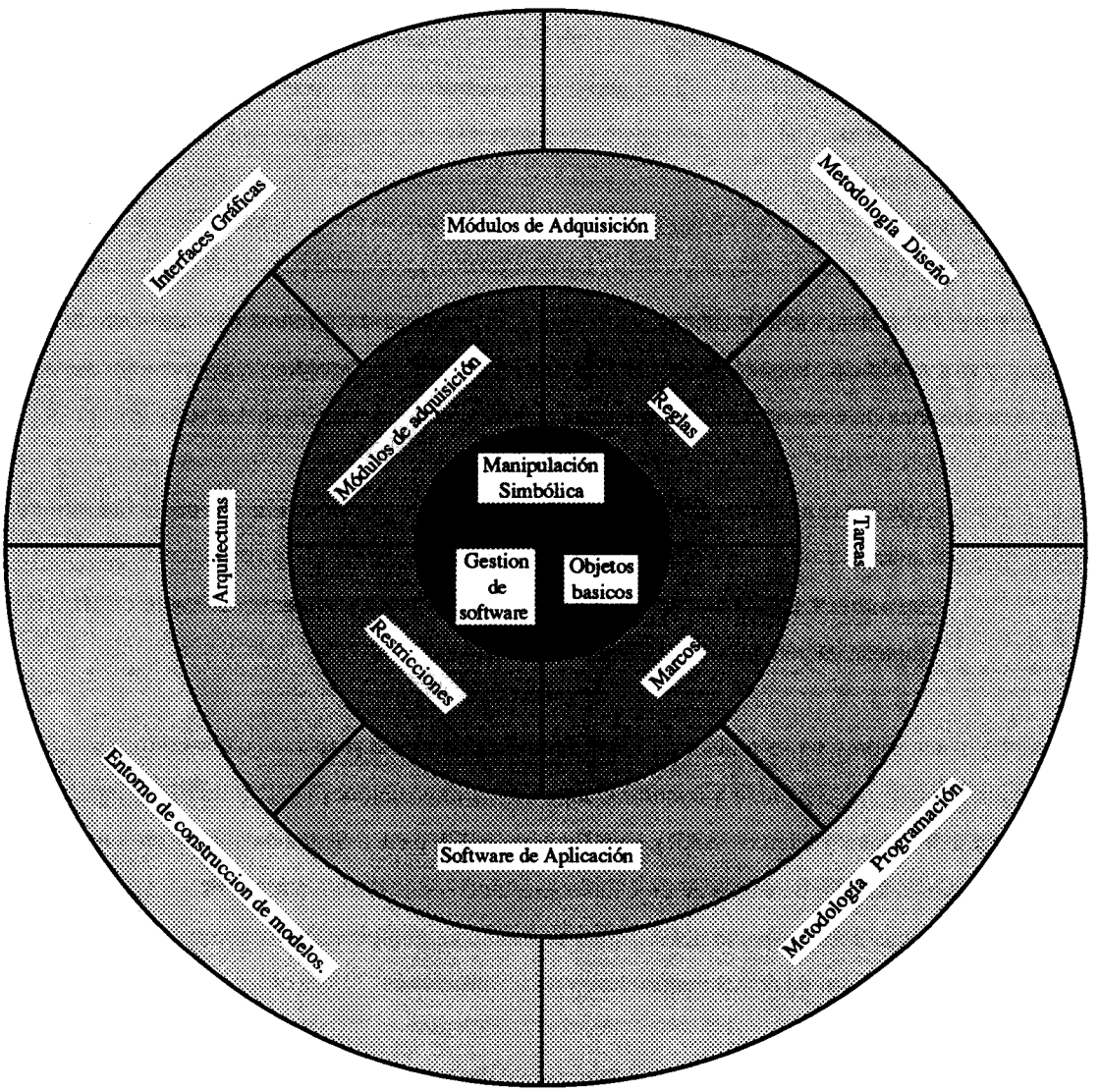


Figura 6o: Cronología del desarrollo de software.

El software básico comprende la implantación del conjunto mínimo de habilidades para poder realizar programación orientada a objetos más una colección de objetos de soporte. Entre las primeras cabe resaltar:

1) Capacidad para manejo simbólico de información (primitivas estilo LISP).
2) Características avanzadas de objetos: polimorfismo y herencia estática y dinámica, paso de mensajes, etc.

y como objetos (o tipos abstractos) más característicos:

- El objeto **Atomo.**
- **Listas.**
- **Colas.**
- **Grafos.**
- **Cadenas.**
- **Métodos.**
- etc.

El software para la implantación de sistemas básicos de representación e inferencia ha significado la creación de un lenguaje (y su módulo de adquisición) de definición del conocimiento mediante reglas, objetos estructurados y sistemas de satisfacción de restricciones, más una colección de objetos entre los que destacan:

- **Objetos, Atributos, Valores** y **Reglas**, para representación del conocimiento en reglas y marcos.
- **Espacios Cualitativos, Variables cualitativas** y **Restricciones cualitativas**, para representación del conocimiento en restricciones.

Los desarrollos anteriores no se realizaron exclusivamente para este proyecto y, por tanto, no se describen en detalle en esta memoria. Se pueden encontrar magnificas exposiciones en [Molina 90], para el Sistema de Administración de Reglas y Marcos (SAR), y en [Izquierdo 92], para el Sistema de Satisfacción de Restricciones (SSR). Asimismo, el software de aplicación e interfaces tampoco se describe por no ser relevante desde el punto de vista de la investigación. Respecto del software avanzado para implantación de tareas, en el apartado 6.4.2 se presenta un resumen del diseño de los principales objetos identificados.

Finalmente, en el apartado 6.4.3 se realiza una descripción del módulo reconocedor del lenguaje de Definición del Conocimiento (DECON). Este módulo no admitía un diseño orientado a objetos por limitaciones de las herramientas de soporte.

6.4.2. Software para implementación de tareas.

Este apartado tiene por objetivo efectuar una revisión técnica del diseño de software realizado, como soporte de las tareas. Esta revisión no pretende ser exhaustiva, por razones obvias de espacio, y se limitará, en consecuencia, a los elementos más relevantes del diseño. En el Laboratorio de Sistemas Inteligentes existe una descripción detallada del total de módulos desarrollados.

Dada la metodología de diseño y programación seguida, es inevitable que esta revisión técnica posea también la típica organización por objetos, con el siguiente índice por objeto:

1) Objetivo de diseño: funcionalidad principal requerida por la cual tiene lugar la creación del objeto.
2) Relaciones: ubicación del objeto dentro de la jerarquía general, estableciendo sus mecanismos de herencia, si los hubiere, así como sus relaciones estructurales (de instancia, de subclase, etc) con otros tipos de objetos.
3) Declaración: definición de la estructura del objeto, identificando sus componentes públicas y privadas. El lenguaje de declaración a utilizar es el lenguaje C.
4) Funcionalidad: métodos principales entre los que conforman la visión externa del objeto. Por cada método, se detalla su algoritmo en lenguaje natural. Los métodos básicos (creación y borrado de clases e instancias, métodos de acceso, etc) no se describen, a excepción de su mecanismo de herencia si es de tipo particular.

Como es sabido, en ocasiones el diseño de un objeto provoca de forma natural la aparición de algún otro objeto derivado. Cuando sea este el caso, los objetos derivados se describirán de forma resumida en el mismo apartado que su objeto principal.

6.4.2.1. Tarea General de Control.

Objetivo.

La TGC es el objeto donde reside la estrategia principal; es decir, la base de conocimiento de control general y su procedimiento de interpretación. Este procedimiento se desglosa fundamentalmente en la evaluación, síntesis y revisión del árbol de hipótesis de problemas, así como su verificación.

Relaciones.

La TGC es un objeto único y aislado para cada entorno que se construya sobre la arquitectura. Por tanto, no se incluye en ninguna jerarquía ni presenta esquema de herencia.

Declaración.

Su estructura física es la siguiente:

```
typedef struct {
st_atomo        Atm;            /* Extensión de átomo. Pudiendo recibir mensajes. */
```

```
lista       Arbol;      /* Lista de hipótesis que constituyen las raíces del árbol.
                           Estas hipótesis son instaladas en tiempo de carga por el
                           módulo reconocedor de DECON. */
lista       Candidatas; /* Lista de hipótesis candidatas a ser entrada del
                           simulador. */
objeto      Selección;  /* Hipótesis elegida, entre las candidatas, para su entrada
                           en el simulador. */
} st_tgc, *tgc;  /* Tarea General de Control. */
```

Toda la estructura de la tarea es privada, comunicándose esta con el exterior únicamente a través de sus métodos externos.

Funcionalidad.

Evaluación de una hipótesis:

Propósito: Evalúa una hipótesis, dando valor a su nivel de activación y al de sus descendientes. Devuelve la hipótesis evaluada.

Efecto lateral: Se modifica la lista de hipótesis candidatas.

Algoritmo:

- *Sea Hip la hipótesis a evaluar.*
- *Evaluar el nivel de activación de Hip.*
- *Si (Hip) Nivel de activación ES No:*
  - *Para toda subhipótesis Sh de Hip:*
    - *Negar Sh (asignar No a su nivel de activación).*
- *Si (Hip) Nivel de activación ES Posible:*
  - *Para toda subhipótesis Sh de Hip:*
    - *Negar Sh (asignar No a su nivel de activación).*
- *Si (Hip) Nivel de activación ES Si:*
  - *Llevar Hip a la lista de hipótesis candidatas.*
- *Devolver Hip.*

Revisión de una hipótesis:

Propósito: Revisa una hipótesis, cambiando su nivel de activación y evaluando sus hipótesis descendientes. Devuelve la hipótesis revisada.

Algoritmo:

- *Sea Hip la hipótesis a revisar.*
- *Sea LPD la lista de patrones descendientes de Hip.*

- *Para todo patrón Ptr de LPD:*
  - * *Olvidar la verificación actual de Ptr.*
  - * *Verificar Ptr.*
- *Devolver Hip.*

Negación de una hipótesis:

Propósito: Niega la hipótesis y todas sus descendientes.

Algoritmo:

- *Sea Hip la hipótesis a negar.*
- *Si (Hip) Nivel de activación ES No: no hacer nada.*
- *Si (Hip) Nivel de activación ES Si:*
  - * *Asignar No al Nivel de activación de Hip.*
  - * *Quitar Hip de la lista de hipótesis candidatas de la tarea.*
- *Si (Hip) Nivel de activación ES Posible:*
  - * *Asignar No al Nivel de activación de Hip.*
- *Para toda subhipótesis Sh de Hip:*
  - * *Negar Sh (recursión).*
- *Devolver Hip.*

Evaluación del árbol de hipótesis:

Propósito: Evalúa el árbol de hipótesis de la tarea, generando la lista de candidatas a ser entrada del simulador. Devuelve la lista generada.

Algoritmo:

- *Para toda hipótesis R, raíz del árbol de hipótesis de la tarea:*
  - * *Evaluar la hipótesis R.*
- *Devolver la lista de hipótesis candidatas.*

Selección de hipótesis para simulación:

Propósito: Revisa la lista de hipótesis candidatas de la tarea, un vez verificada, por simulación, al menos una hipótesis. Como resultado se elige una nueva hipótesis. Devuelve la nueva hipótesis seleccionada.

Efecto lateral: La lista de hipótesis candidatas puede sufrir cambios, ya que los patrones descendientes de la hipótesis previamente verificada se desactivan.

Algoritmo:

- *Sea Hp la hipótesis previamente verificada.*
- *Sea LPD la lista de patrones descendientes de Hp.*

- *Para todo patrón Ptr de LPD que haya sido verificado:*
  - * *Desactivar Ptr.*
- *Para toda hipótesis de la lista de candidatas de la tarea:*
  - * *Olvidar su nivel de activación.*
- *Sea LF la nueva lista de hipótesis candidatas (inicialmente LPD).*
- *Restar Hp a LF.*
- *Para toda hipótesis Hip de LF:*
  - * *Evaluar Hip.*
- *Sea Hn la nueva hipótesis seleccionada para el simulador.*
- *Sea LPn la lista de patrones descendientes de Hn.*
- *Para todo patrón Pn representante de Hn:*
  - * *Activar su intervención en tareas de generación de escenarios.*
- *Sea D el conjunto diferencia entre el total de patrones y los representantes de Hn.*
- *Para todo patrón Pd del dominio D:*
  - * *Desactivar su intervención en tareas de generación de escenarios.*
- *Para todo patrón de LPn:*
  - * *Olvidar su estado de verificación (posiblemente se volverán a verificar).*
- *Devolver Hn.*

Hipótesis de exploración.

Objetivo.

Una hipótesis se define como la asociación de un conjunto de patrones de comportamiento final cuya existencia se desea investigar en simulación. Toda hipótesis supone una abstracción sobre el conjunto de patrones que de ella descienden, ya que solo alberga en ella a un subconjunto de estos que actúa como representante.

Relaciones.

La hipótesis se define como un marco clásico (sin estructura propia) y, por tanto, se sitúa en la jerarquía como subclase de la clase general de objetos. En cuanto a su propia linea jerárquica, ésta se forma en principio a un solo nivel, con un concepto de clase y espacio para una colección de instancias. Su definición como marco posibilita la existencia de conceptos intermedios (subclases), como se muestra en la figura 6p.

Su esquema de herencia es el común de los marcos. En cuanto a sus relaciones funcionales (paso de mensajes), estas se producen con la Tarea General de Control.

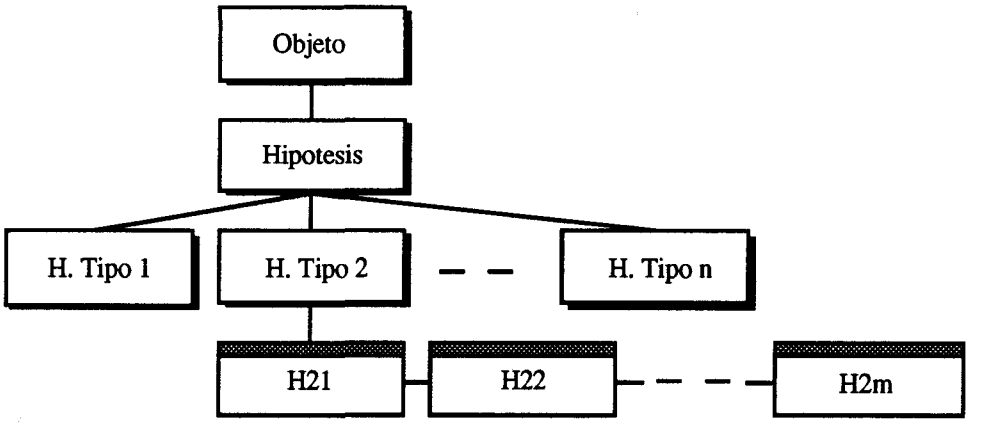


Figura 6p: Jerarquía de hipótesis de exploración.

Declaración.

La hipótesis no tiene estructura propia. Su estructura es la de un marco y la definición general de la clase se realiza vía DECON, como subclase de la clase general de los objetos, según se muestra en la figura 6q.

```
OBJETO: Hipotesis de exploracion SUBCLASE DE Objeto.
   VARIABLES: Subhipotesis de (INSTANCIAS DE Hipotesisde exploracion] [Constante];
              Representantes (INSTANCIAS DE Patron de comportamiento final) [Constante];
              Nivel de activacion (Si, No, Posible) = Posible [F_Nivel de activacion, ()];
              Interes (NUMERICO) [F_Interes, ()];
              Area de exploracion (INSTANCIAS DE TBS) [Reglas]
```

Figura 6q: Definición de la clase general Hipótesis de exploración.

Funcionalidad.

Cálculo del grado de interés de una hipótesis:

Propósito: Devuelve el grado de interés de una hipótesis (para su entrada en simulación) en función del interés de sus patrones representantes. Esta función es la que se aplica por defecto, cuando no hay criterio de usuario definido mediante base de conocimiento.

Algoritmo:

- *Sea Hip la hipótesis cuyo grado de interés se desea calcular.*
- *Sea I el interés de la hipótesis (inicialmente 0).*
- *Para todo patrón Ph representante de Hip:*
  * *Sumar a I el grado de interés de Ph.*
- *Devolver I.*

Cálculo del nivel de activación de una hipótesis:

Propósito: Devuelve el nivel de activación de una hipótesis.

Algoritmo: por defecto (redefinible por el usuario) se deduce mediante las reglas DECON siguientes (por defecto No activo):

- *PARA TODA ?h Hipótesis*
  SI (?h) Patrones representantes ES ?p,
  (?h) Patrones representantes ES ?q,
  (?p) Identidad NO ES ?q,
  (?p) Nivel de activación ES Activo,
  (?q) Nivel de activación NO ES Activo
  ENTONCES [1] (?h) Nivel de activación ES Posible,
  -> (?h), Desasigna! {Nivel de activación, Activo},
  -> (?h), Desasigna! {Nivel de activación, No activo}
- *PARA TODA ?h Hipótesis*
  SI (?h) Patrones representantes ES ?p,
  (?p) Nivel de activación ES Activo,
  ENTONCES [1] (?h) Nivel de activación ES Activo.
  -> (?h), Desasigna! {Nivel de activación, No activo}

Cálculo del foco de exploración de una hipótesis:

Propósito: Devuelve la hipótesis con su área de exploración calculada como la unión de las áreas de influencia de sus patrones representantes. Un área de influencia es un conjunto de TBSs.

Algoritmo: por defecto (redefinible por el usuario) se deduce mediante la regla DECON siguiente:

- *PARA TODA ?h Hipótesis*
  SI (?h) Patrones representantes ES ?p,
  (?p) Area de influencia ES ?Tbs,
  ENTONCES (?h) Area de exploración ES ?Tbs

Patrones de comportamiento final.

Objetivo.

Un patrón de comportamiento final se define como un objeto para representar un esquema tipo de situación problemática, hacia la verificación de la cual se debe dirigir el razonamiento en predicción. Básicamente, un patrón debe actuar como elemento de control de una variable de estado del sistema.

Relaciones.

De forma similar al objeto Hipótesis ya descrito, el patrón se define como extensión de la clase general de objetos (es decir, como un marco) sin estructura propia. Se sitúa en la jerarquía, por tanto, como subclase de aquella. El usuario puede definir conceptos intermedios, definitorios de subclases de patrones, según el esquema mostrado en la figura 6r.

Declaración.

El patrón no tiene estructura propia. Su estructura es la de un marco y la definición general de la clase se realiza vía DECON, como subclase de la clase general de los objetos, según se muestra en la figura 6s.

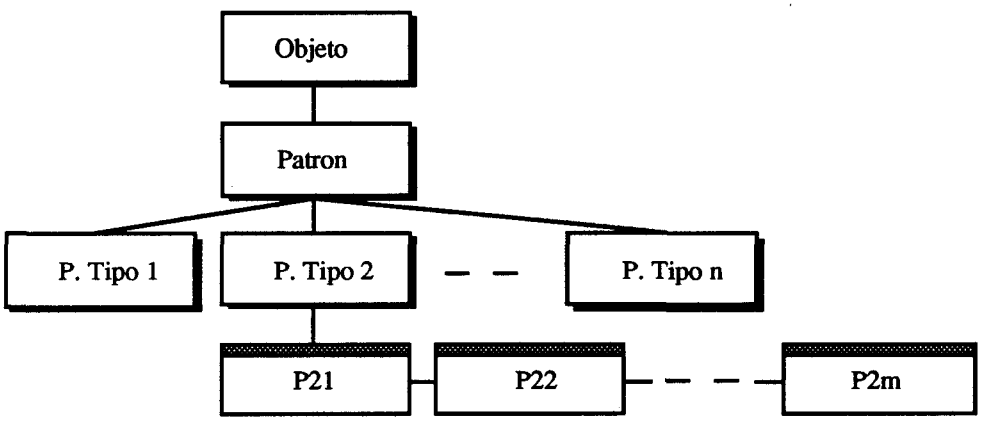


Figura 6r: Jerarquía de patrones de comportamiento final.

```
OBJETO: Patron de comportamiento final SUBCLASE DE Objeto
  VARIABLES: Interviene (Si, No) = Si [Constante];
             Activo (Si, No) = Si [Reglas];
             Verificado (Si, No) = No [Reglas];
             Valoracion Actual (INSTANCIAS DE Niveles] = Normalidad [Reglas];
             Valoracion Prevista [INSTANCIAS DE Niveles] = Normalidad [Reglas];
             Gradiente de cambio (INSTANCIAS DE Gradientes) = Nulo [Reglas];
             Superficie (NUMERICO) [Constante];
             Interes (NUMERICO) [F_Interes, ()];
             Area de influencia (INSTANCIAS DE TBS) [Constante]
```

Figura 6s: Definición de la clase general Patrón de comportamiento final.

Funcionalidad.

Como se puede apreciar en su declaración, la funcionalidad general del patrón se instrumenta mediante bases de conocimiento formuladas en reglas. Unicamente el cálculo del grado de interés de un patrón se realiza mediante función simple, que también puede ser redefinido en cualquiera de las subclases o instancias de patrón que el usuario decida crear.

6.4.2.2. Tareas Básicas de Simulación.

Objetivo.

Las TBSs tienen por objeto representar adecuadamente la respuesta de los objetos físicos que componen el sistema de flujo. La TBS debe albergar el conocimiento necesario para relacionar las entradas al componente con su predicción de salidas.

Relaciones.

La TBS se ubica en la jerarquía como subclase de la clase general de los objetos. Esto le permite heredar sus características como marco. Alguno de sus atributos, en concreto sus entradas y salidas, definen al mismo tiempo relaciones con otros objetos (TBSs y TGEs). El conjunto de esas relaciones conforma la topología del sistema objeto de modelización. La figura 6t muestra como la TBS se inscribe en la jerarquía de objetos, suministrándose inicialmente dos subclases de TBS, que el usuario puede extender convenientemente. La extensiones mostradas en la figura corresponden a clases definidas para el entorno hidrológico (ver anejo A).

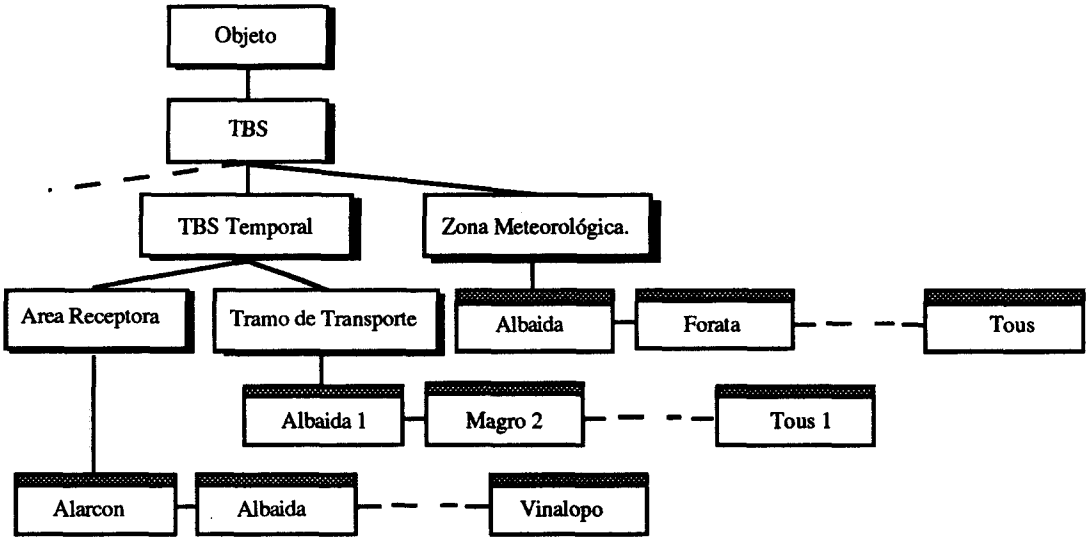


Figura 6t: Jerarquía de Tareas Básicas de Simulación (dominio hidrológico).

La TBS tiene un esquema de herencia propio, extensión del mecanismo general de herencia entre objetos. Sus principales características se describen en el apartado de funcionalidad.

Declaración.

Su estructura física es la siguiente:

```
typedef struct {
st_objeto     Obj;          /* Extensión de objeto, integrada explícitamente en la
                               jerarquía. */
lista         Propiedades;  /* Atributos que representan características físicas
                               constantes, cuyo valor se define en tiempo de carga. */
lista         Entradas;     /* Variables de entrada de la TBS.*/
lista         Internas;     /* Variables internas de proceso.*/
lista         Salidas;      /* Variables de salida de la TBS.*/
lista         Estado;       /* Variables internas o de salida cuyo comportamiento es
                               de tipo reactivo y, por tanto, requerirán memoria de su
                               estado a lo largo del tiempo.*/
lista         Creación;     /* Bases de conocimiento (reglas y restricciones) que
                               actúa en tiempo de carga del modelo, para inferencia de
                               algunas variables internas .*/
lista         Inicio;       /* Bases para la deducción de estados iniciales de
                               simulación.*/
lista         Proceso;      /* Bases alternativas de proceso sometidas a la estrategia
                               de control. */
lista         Final;        /* Bases de asignación de variables de salida..*/
lista         LRCreación;   /* Linea de razonamiento en tiempo de creación (para
                               borrados de consulta: uso interno).*/
lista         LRInicio;     /* Linea de razonamiento en deducción de estados
                               iniciales (para borrados de consulta: uso interno).*/
lista         BConsulta;    /* Variables afectadas por borrado de consulta en un paso
                               de simulación (uso interno).*/
char          dt;           /* Intervalo de transición: periodo de tiempo entre pasos
                               de simulación. */
} st_tbs, *tbs;   /* Tarea Básica de Simulación. */
```

Toda la estructura es privada a la TBS, excepto las variables de entrada y salida.

Funcionalidad.

La funcionalidad principal de una TBS gira entorno a su estrategia general y a los mecanismos de herencia. La estrategia general que aquí se muestra es la que se suministra por defecto, pudiendo ser redefinida por usuario.

Deducción de la respuesta de una TBS:

Propósito: Devuelve la predicción de respuestas de la TBS a partir de sus solicitaciones.

Algoritmo:

- *1) Fase de preparación.*
  - * *Sea Ts el tiempo de simulación (inicialmente 0).*
  - * *Para toda solicitación S de la lista de entradas de la TBS:*
    - * *Evaluar S.*
  - * *Evaluar Bases de Inicio (deducción de estados iniciales) de forma análoga al paso 2.2.*
- *2) Fase de Proceso.*
  - * *2.1) Evaluar Base de Control, obteniendo lista de bases para proceso P (la base de control es la primera base de proceso). Si P es vacía, ir a 3.*
  - * *2.2) Para toda base B del conjunto P:*
    - * *Si B es un sistema de restricciones:*
      - * *Convertir variables de B a formato de restricciones (interfaz SAR-SSR).*
      - * *Evaluar B.*
      - * *Convertir variables de B a formato de SAR (interfaz SSR-SAR).*
    - * *Si B es una base de reglas:*
    - * * *Evaluar B.*
  - * *2.3) Incrementar tiempo de simulación Ts según intervalo de transición (dt).*
  - * *2.4) Ir a 2.1.*
- *3) Fase de finalización.*
  - * *Evaluar Bases de finalización (deducción de salidas) de forma análoga al paso 2.2.*
- *4) Devolver la TBS.*

Creación de una instancia de TBS:

Propósito: Devuelve una nueva instancia de TBS, heredando de una clase dada.

Algoritmo:

- *Sea I la nueva instancia (inicialmente NIL).*
- *Sea C la clase de TBS de la cual se crea la instancia.*
- *Reservar memoria para la estructura de I.*
- *Heredar desde C hacia I (herencia estática).*
- *Instalar I como instancia de C.*
- *Devolver I.*

Esquema de herencia para una instancia TBS:

Propósito: Realiza la herencia estática de una instancia de TBS, a partir de una clase dada. Devuelve la instancia de TBS.

Algoritmo:

- *Heredar como átomo (paso de mensajes).*
- *Heredar Intervalo e Transición (dt).*
- *Heredar Entradas.*
- *Heredar Salidas.*
- *Heredar lista de Propiedades.*
- *Heredar lista de Variables Internas (propiedades deducidas y variables de estado).*
- *Heredar Bases de Creación.*
- *Heredar Bases de Inicio.*
- *Heredar Bases de Proceso.*
- *Heredar Bases de Finalización.*
- *Devolver la instancia de TBS.*

Herencia de una base de conocimiento formulada en reglas:

Propósito: Realiza la herencia estática de una base de TBS formulada en reglas de inferencia. Nótese como las reglas no forman parte del esquema de herencia. Esto es porque se heredan dinámicamente en tiempo de ejecución.

Algoritmo:

- *Sea BC la base de conocimiento de la clase de TBS de la cual se hereda.*
- *Sea BI la nueva base de conocimiento (instancia a definir).*
- *Reservar memoria para BI.*
- *Para todo objetivo O de la base BC:*
  - * *Crear un nuevo objetivo I como instancia de O.*
  - * *Asociar el nuevo objetivo I a la base BI.*
- *Devolver la nueva base BI.*

Herencia de una base de conocimiento formulada en restricciones:

Propósito: Realiza la herencia estática de una sistema de satisfacción de restricciones (SSR).

Algoritmo:

- *Sea SSR-C la base de conocimiento en restricciones de la clase de TBS de la cual se hereda.*

- *Sea SSR-I la nueva base de conocimiento en restricciones (instancia a definir).*
- *Sea TBS-I la instancia de TBS a la que asociar el nuevo sistema de restricciones.*
- *Reservar memoria para BI.*
- *Para toda restricción R de la base SSR-C:*
    - * *Para todo operando O de R:*
        - * *Sea V la variable asociada al operando O.*
        - * *Si existe en TBS-I una instancia VI de V:*
            - * *Asociar VI como operando para una nueva restricción.*
        - * *Si no existe en TBS-I una instancia VI de V:*
            - * *Asociar V como operando para una nueva restricción.*
    - * *Definir una nueva restricción RI como instancia de R, con los operandos obtenidos y la operación de R.*
    - * *Agregar RI como restricción de SSR-I.*
- *Devolver SSR-I.*

6.4.2.3. Tareas de Generación de Escenarios.

Objetivo.

Las TGEs tienen por objeto relacionar los escenarios de salida de grupos de TBSs con los escenarios de entrada de otros grupos de TBSs en niveles consecutivos de simulación. La TGE debe albergar el conocimiento necesario para filtrar dichos escenarios, reduciendo la combinatoria de influencias de unos componentes físicos sobre otros en el sistema de flujo.

Relaciones.

La TGE se define como subclase de la clase general de los objetos, extendiendo la estructura de esta. Posee, por tanto, relaciones estructurales de subclase e instancia. Sin embargo, en este caso no se admite la definición de conceptos intermedios (subclases del concepto general) vía DECON, sino únicamente instancias de las clases de tarea que habrán sido creadas, como parte del entorno, por software. Este esquema de implementación era más simple y se eligió por limitaciones del tiempo de desarrollo disponible. La figura 6u muestra el esquema jerárquico, para las TGEs, creado para el entorno hidrológico mostrado en el anejo A.

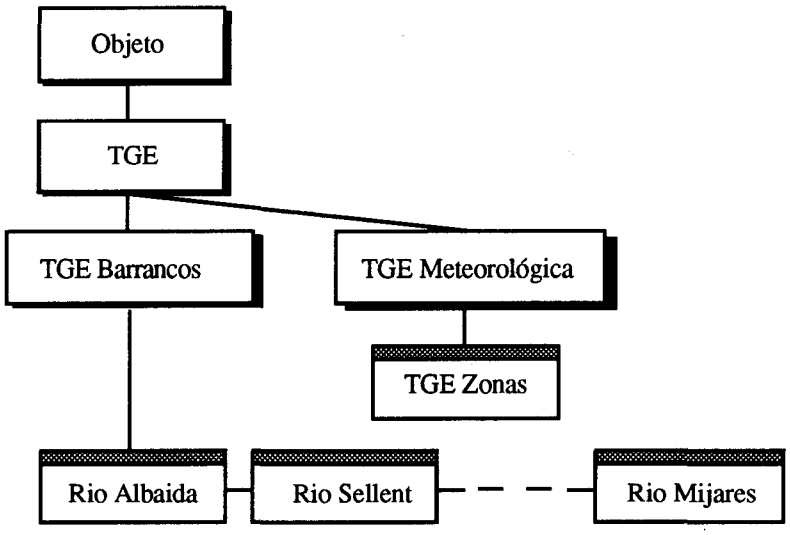


Figura 6u: Jerarquía de Tareas de Generación de Escenarios.

Declaración.

Su estructura física es la siguiente:

```
typedef struct {
st_objeto       Obj;                /* Extensión de objeto, integrada explícitamente en la
                                       jerarquía. */
lista           DominiosEntrada;    /* Dominios de respuesta de las TBSs entrantes a la
                                       TGE. */
lista           PatronReferencial;  /* Valor de referencia para cada dominio de entrada. */
lista           Pertinencia;        /* Conjunto de bases de reglas y/o restricciones que
                                       aplican los criterios de pertinencia. */
lista           Verosimilitud;      /* Conjunto de bases de reglas y/o restricciones que
                                       aplican los criterios de verosimilitud. */
ssr             Sintesis;           /* Base de restricciones de síntesis, unión de las bases de
                                       restricciones de Pertinencia y Verosimilitud. */
lista           EscenarioRespuesta; /* Asignación individual de variables al
                                       subconjunto correspondiente de los dominios de
                                       entrada, para actuar como entrada de TBSs en el
                                       siguiente nivel de simulación. */
} st_tge, *tge;
```

Derivado de su condición de marco, las TGEs definidas por el usuario como instancias del concepto general podrán llevar variables propias adicionales, a ser utilizadas en cada una de las bases de conocimiento. En el anejo A se muestran ejemplos de definición de instancias de TGE.

Funcionalidad.

Su funcionalidad principal gira entorno a la estrategia general de la tarea y a su esquema de herencia. Este último, además, se resuelve de forma similar al caso de las TBSs, por lo que no se incluye.

Estrategia general de ejecución de una TGE:

Propósito: Genera un escenario respuesta por filtrado de los dominios de entrada, aplicando los criterios de pertinencia por objetivos y verosimilitud contenidos en las bases de conocimiento de la tarea.

Algoritmo:

- *Evaluar linea de Pertinencia: obtención de subdominios pertinentes.*
- *Si alguna base de pertinencia resuelve infactibilidad:*
  - * *Terminar: no existe escenario dentro de la hipótesis de exploración considerada.*
- *Evaluar linea de Verosimilitud: obtención de subdominios verosímiles.*
- *Si alguna base de verosimilitud resuelve infactibilidad:*
- * *Terminar: no existe escenario que, siendo factible según la hipótesis de exploración propuesta, sea verosímil.*
- *Evaluar la base de síntesis: obtención de subdominios pertinentes y verosímiles..*
- *Si la base de síntesis resuelve infactibilidad:*
- * *Terminar: no existe escenario según los criterios de Pertinencia y Verosimilitud aplicados conjuntamente.*
- *Generación del escenario respuesta; obtención del escenario más cercano al patrón de referencia:*
  - * *Aplicar el algoritmo DS (ver capítulo 8.2).*
- *Devolver el escenario obtenido o NIL si no se obtuvo ninguno.*

6.4.3. Módulos reconocedores del lenguaje de DEfinición del CONocimiento (DECON).

La carga del conocimiento por parte del sistema se realiza a niveles diferentes:

1) La parte más alta de la jerarquía, es decir, la clases generales y las subclases de más alto nivel se encuentran programadas, por razones de eficiencia; así, por ejemplo, ocurre con la clase general de objetos, atributos, valores o la clase de las tareas básicas de simulación, entre otras. Esto significa que

solo se asumen por software los objetos componentes de la arquitectura, lo cual confiere a esta la generalidad pretendida.

2) El resto de las subclases se comunican al sistema en forma de ficheros de texto con la descripción del conocimiento. Este conocimiento es suministrado por los constructores del entorno y está descrito en el lenguaje de representación DECON.
3) La información sobre subclases de más bajo nivel y sobre las instancias constituye también información de entrada y es suministrada, en este caso, por expertos en las parcelas del mundo real que se quieren representar.

Esta estructura de niveles permite dotar a la parte baja de la jerarquía de objetos de gran flexibilidad, ya que sobre una base general de clases de objetos se pueden desarrollar estructuras diferentes, sin necesidad de recompilar programa alguno. La construcción del primer nivel es una tarea de programación. La edición de unidades cognitivas se realiza mediante la redacción de ficheros de texto que describan los objetos y sus relaciones. El problema que se plantea es el de realizar un módulo cargador que tome ficheros de texto con descripciones de objetos y los traduzca a estructuras procesables en memoria. Se tienen las siguientes especificaciones adicionales:

1) Las descripciones de los objetos y las relaciones entre ellos están en ficheros de texto. El lenguaje empleado debe ser lo más cercano posible al lenguaje natural, con una sintaxis cómoda y flexible. En concreto, se respetará la posibilidad de que los nombres de las entidades consten de varias palabras separadas por blancos.
2) La sintaxis de los lenguajes de descripción de las diferentes entidades ha de tener una estructura uniforme. Atributos similares de objetos distintos se han de poder describir de la misma forma. También ha de habilitarse un sistema por el cual la información pueda distribuirse en diferentes ficheros de tamaño manejable (con un mecanismo del tipo *include* similar al que disponen muchos lenguajes de programación). Los ficheros con la descripción del conocimiento pueden contener errores (pues son editados manualmente por ingenieros del conocimiento o por usuarios) y el comportamiento del sistema ante ellos debe ser el siguiente:
   - Informar al usuario de la naturaleza y localización del error.
   - Proseguir el reconocimiento, de forma que las repercusiones del error sean mínimas.
3) El reconocedor debe construirse de forma modular, para que pueda usarse en sistemas diferentes, incluyendo en cada módulo solamente la capacidad de reconocer los objetos que cada sistema necesite.

Dadas las especificaciones anteriores y al realizarse el desarrollo sobre el sistema operativo Unix se decide como aconsejable utilizar las herramientas *yacc* y *lex* , que sirven respectivamente para construir analizadores sintácticos y léxicos.

6.4.3.1. Función de carga.

Para cargar estructuras en memoria a partir de ficheros de texto con la descripción del conocimiento se ha construido una función, con ayuda de yacc y lex, que recibe como parámetro el nombre del fichero que contiene el conocimiento y que, a partir de él, construye una estructura en memoria que representa, fielmente, el conocimiento descrito. Esta función es única y es capaz de reconocer todos los objetos del sistema que se carguen de esta forma (objetos, atributos, valores, reglas, restricciones, etc.). Puede ser llamada varias veces, añadiéndose cada vez nuevas estructuras sobre la estructura en memoria ya existente. Finalmente, devuelve un valor que indica si la carga ha sido satisfactoria o no, dependiendo de los errores encontrados o de otros posibles problemas (corrupción de los ficheros, falta de espacio en memoria, etc.).

6.4.3.2. Ficheros para la descripción del conocimiento.

Los ficheros de datos están constituidos por descripciones de las unidades cognitivas del modelo. En cada unidad de descripción se detalla una subclase o instancia, por ejemplo es una unidad de descripción la que declara la clase de los Patrones de Comportamiento Final o la que declara una instancia, específica, de dicha clase. Su estructura general es la siguiente:

1) Etiqueta de tipo de objeto o entidad (p. ej.: "OBJETO:", "TBS:"; en el caso de las reglas, es la palabra "SI" la que hace las veces de etiqueta de regla).
2) Nombre de la instancia o subclase (en las reglas no hay).
3) En algunos objetos, nombre de la clase antecedente en la jerarquía.
4) Atributos.
5) Resto del conocimiento, con una estructura análoga a la ya comentada.
6) En algunos casos es preciso usar una etiqueta de fin de declaración de objeto o entidad.

Dentro de un fichero de descripción del conocimiento pueden aparecer referencias a otros ficheros, de forma que cuando el reconocedor llegue al punto en que se encuentra alguna de estas referencias, continúe el reconocimiento en el fichero indicado y regrese al mismo lugar cuando lo termine. Este mecanismo permite repartir el conocimiento entre tantos ficheros como sea necesario y ordenarlos en una jerarquía de un número ilimitado de niveles. Dentro de un mismo fichero puede haber descripciones de objetos de distinto tipo y llamadas a fichero subordinados. Obviamente, estas llamadas no pueden estar dentro de una unidad de descripción, sino situadas entre ellas, puede considerarse una norma de buen estilo el situar estas referencias a otros ficheros al comienzo de cada fichero, como se puede comprobar en los ejemplos que se presentan en este informe.

6.4.3.3. Lenguajes.

Los lenguajes de descripción de objetos o entidades que constituyen una representación o modelo están basados en palabras reservadas, que sirven como etiquetas de identificación de los nombres de objetos que se escriben a continuación de ellas. Por ejemplo, la descripción de una instancia de la entidad *objeto* tiene la siguiente estructura:

OBJETO: <nombre de instancia> **ES UN** <nombre de clase>
CONSTANTES:
<lista de constantes>
VARIABLES:
<lista de variables>

Una de las características más peculiares del DECON, que figura como requerimiento, es la posibilidad de emplear nombres formados por varias palabras. Las únicas restricciones para la construcción de nombres son:

1) No superar una longitud determinada (actualmente, 80 caracteres).
2) Que su primera palabra comience con una letra.
3) Que no incluyan palabras o símbolos reservados. Las palabras reservadas pueden formar parte de un nombre si no van totalmente en mayúsculas o si forman parte de una palabra más larga.

6.4.3.4. Analizadores.

La estructura interna del reconocedor es la misma que la de un compilador convencional. Esta compuesto por un analizador léxico o *scanner* y un analizador sintáctico o *parser*. Este último es el que lleva el control del reconocimiento y emplea el analizador léxico como una función a la que le pide *tokens* de uno en uno.

El analizador léxico lee el fichero de entrada hasta reconocer un token que devolverá al analizador sintáctico. Como acción semántica asociada al reconocimiento de un token, almacena cierta información sobre el mismo en una estructura llamada *símbolo*, que hace las funciones de la clásica tabla de símbolos de un compilador convencional.

El analizador sintáctico reconoce la secuencia de tokens que recibe de acuerdo con la gramática del lenguaje. Asociadas a las reglas de esta gramática se procesan rutinas semánticas para realizar las siguientes funciones:

1) Creación de la estructura procesable de memoria especificada en el fichero de descripción del conocimiento.
2) Control de corrección semántica del conocimiento.
3) Tratamiento de errores: presentación de mensajes y recuperación (en la medida de lo posible) del error producido.
4) Cambio de fichero, con ayuda de una pila: paso a ficheros subordinados y regreso al fichero llamador al recibir un token de fin de fichero.

El funcionamiento del analizador sintáctico es ascendente, es decir, reconoce primero las metanociones cuya derivación se compone sólo de tokens y va ascendiendo por la jerarquía de reglas hasta que tiene todas las metanociones que componen la derivación del axioma. Ejemplo de análisis ascendente:

Gramática:

Axioma	:	Elemento
	|	Axioma Elemento
Elemento	:	TOKEN_1 TOKEN_2

Tira que hay que reconocer (suministrada token a token por el analizador léxico):

===> TOKEN_1 TOKEN_2 TOKEN_1 TOKEN_2

Secuencia de reglas aplicada:

Elemento: TOKEN_1 TOKEN_2
===> Elemento TOKEN_1 TOKEN_2
Axioma: Elemento
===> Axioma TOKEN_1 TOKEN_2
Elemento: TOKEN_1 TOKEN_2
===> Axioma Elemento
Axioma: Axioma Elemento
===> Axioma

6.4.3.5. Tratamiento de errores.

La aparición de un error en los ficheros de descripción del conocimiento provoca la presentación de un mensaje de error mediante un mensaje al objeto *error*. Después se proseguirá el reconocimiento, que no terminará hasta que se agoten los ficheros de datos. De esta forma se pueden obtener en una sola pasada todos los mensajes de error.

Se puede emplear un indicador que se active cuando aparezca un error de determinada gravedad y que haga que la función de carga devuelva el valor de reconocimiento no satisfactorio.

Los errores de los ficheros de descripción del conocimiento son de tres tipos:

a) Errores sintácticos. Son los que se producen al recibir un token que no encaja en ninguna derivación posible. Su recuperación se hace mediante reglas que contienen en su parte izquierda la metanoción"error" (estas reglas son transparentes a efectos del usuario). Cuando se produce un error en el reconocimiento de una derivación, en lugar de darse por reconocido su antecedente se considera que se ha encontrado la metanoción *error*. Este efecto se propaga hacia arriba por el árbol de reconocimiento, hasta que se encuentra una regla con la metanoción *error* en su consecuente. En estas reglas se incluyen rutinas semánticas de tratamiento del error. Por ejemplo:

Gramática:

Axioma	:	Lista
Lista	:	Elemento
	|	Lista COMA Elemento
	|	Lista COMA <error>
Elemento	:	TOKEN_1 TOKEN_2

Tira que hay que reconocer (suministrada token a token por el analizador léxico):

===> TOKEN_1 TOKEN_2 COMA TOKEN_2

Secuencia de reglas aplicada:

Elemento : TOKEN_1 TOKEN_2
===> Elemento COMA TOKEN_2
Lista : Elemento
===> Lista COMA TOKEN_2
Elemento : TOKEN_1 TOKEN_2
(No se recibe TOKEN_1, por lo que en lugar de reconocerse Elemento se reconoce <error>)
===> Lista COMA <error>
Lista : Lista COMA <error>
===> Lista
Axioma : Lista
===> Axioma

b) Errores léxicos. Son los que detecta el analizador léxico y pasa al analizador sintáctico en forma de tokens especiales.
c) Errores semánticos. Son los que se detectan en las rutinas semánticas.

Las acciones semánticas asociadas a la detección de un error van encaminadas a suministrar al usuario la información más precisa posible, pero no incluyendo en el mensaje más información de la que se dispone; es decir, el reconocedor comunica lo que encuentra mal, pero no hace suposiciones sobre cual ha sido el error del usuario. El mensaje incluye:

a) Fichero donde se ha encontrado el error.
b) Objeto ya creado al cual se le pueda asociar el error, lo que no siempre será posible (por ejemplo, si hay un símbolo no permitido al principio de un fichero, el error sólo puede asociarse a dicho fichero).
c) Texto que informe de la naturaleza del error: categoría, número de línea dentro del fichero, palabra o símbolo concretos donde el reconocedor ha encontrado el error.
d) Dejar la estructura creada en memoria de la mejor forma posible, teniendo como objetivo principal que un error cometido por el usuario de lugar a un sólo mensaje, y no provoque una serie de errores consecuencia del primero.
e) Actualizar el indicador de reconocimiento no satisfactorio.

6.4.3.6. Organización modular.

El código del reconocedor debe estar organizado de forma que cada aplicación incorpore solamente la parte correspondiente a las entidades que necesita reconocer. Esto sería sencillo si hubiese un método del objeto fichero para cargar cada tipo de entidad. Sin embargo, el empleo de las herramientas *yacc* y *lex* impone la limitación de que haya un solo analizador integrado en cada programa ejecutable. La solución a este problema es que el método de carga sea distinto en cada aplicación. Para ello se ha separado el código de este método en varios ficheros, de forma que para obtener el método de carga de cada aplicación baste con combinar los ficheros adecuados. Se han separado los siguientes componentes del método de carga:

1) La función en C que encapsula la llamada al analizador generado por el *yacc*;
2) La parte del analizador sintáctico común a todos los reconocedores, en forma de código fuente para el *yacc*;
3) Las partes del analizador sintáctico propias del reconocedor de cada tipo de entidad, en forma de código fuente para el *yacc*;
4) La parte del analizador léxico común a todos los reconocedores, en forma de código fuente para el *lex*;

5) Las partes del analizador léxico propias del reconocedor de cada tipo de entidad, en forma de código fuente para el *lex*.

7. EL LENGUAJE DECON COMO SOPORTE A LA CONSTRUCCION DE MODELOS.

Sobre una clase de dominios determinada, el tipo de arquitectura propuesta define un formato tanto para representar el conocimiento como para definir la estrategia de resolución de problemas. La labor de modelización consistirá básicamente en particularizar adecuadamente dichos formatos. Por tanto, la arquitectura sirve como modelo de interpretación del conocimiento sobre el dominio dado y, en consecuencia, como guía para la actividad de ingeniería del conocimiento, en el sentido de la metodología KADS [Breuker, Wielinga 85, 86].

No obstante esto último, no cabe incorporar las tareas presentadas al catálogo de tareas que suministra dicha metodología, por el carácter más específico de aquellas. En [Salmerón 92] se realiza una propuesta en este sentido y es, quizá, optimista en exceso. KADS constituye un intento (muy satisfactorio, por otra parte) de construir un diccionario de soluciones a tipos de problemas, en linea también con las taxonomías de métodos de resolución de problemas de [McDermott 89] o [Clancey 85] y con las tareas genéricas de Chandrasekaran. Pero ese catálogo se construye con independencia de cualquier dominio, aspecto que, si bien en un plano teórico le confieren generalidad, en la práctica de construir sistemas complejos constituye su mayor limitación, pues siguen naufragando en cierta forma al abordar las peculiaridades existentes en el conocimiento de cada rama profesional. Las lineas de razonamiento propuestas en las tareas KADS son válidas y utilizables pero, por su carácter tan general, son precisamente las que se pueden obtener de un experto con relativa facilidad.

A partir de la arquitectura presentada, la construcción de un modelo del conocimiento para simulación de comportamientos en un sistema de flujo requiere los siguientes pasos generales:

1) Formulación de las bases de conocimiento para inferencia de problemas (Tarea de Clasificación).
2) Formulación de la base de conocimiento de control para evaluación de comportamientos generados y de las bases para inferencia de patrones de comportamiento final a explorar (Tarea General de Control).
3) Definición de un conjunto de instancias, de las diferentes clases de Tareas Básicas de Simulación, en representación de los distintos componentes físicos.
4) Definición de las Tareas de Generación de Escenarios, a partir del concepto general, para acuñar criterios de control en los distintos niveles de simulación, en fase de razonamiento profundo.
5) Formulación de las bases de conocimiento para inferencia de criterios de control y recomendación (Tarea de Planificación).

No debe entenderse que las fases anteriores deban abordarse necesariamente con la cronología presentada. A este respecto, el lenguaje de definición del conocimiento (DECON) no impone ninguna limitación.

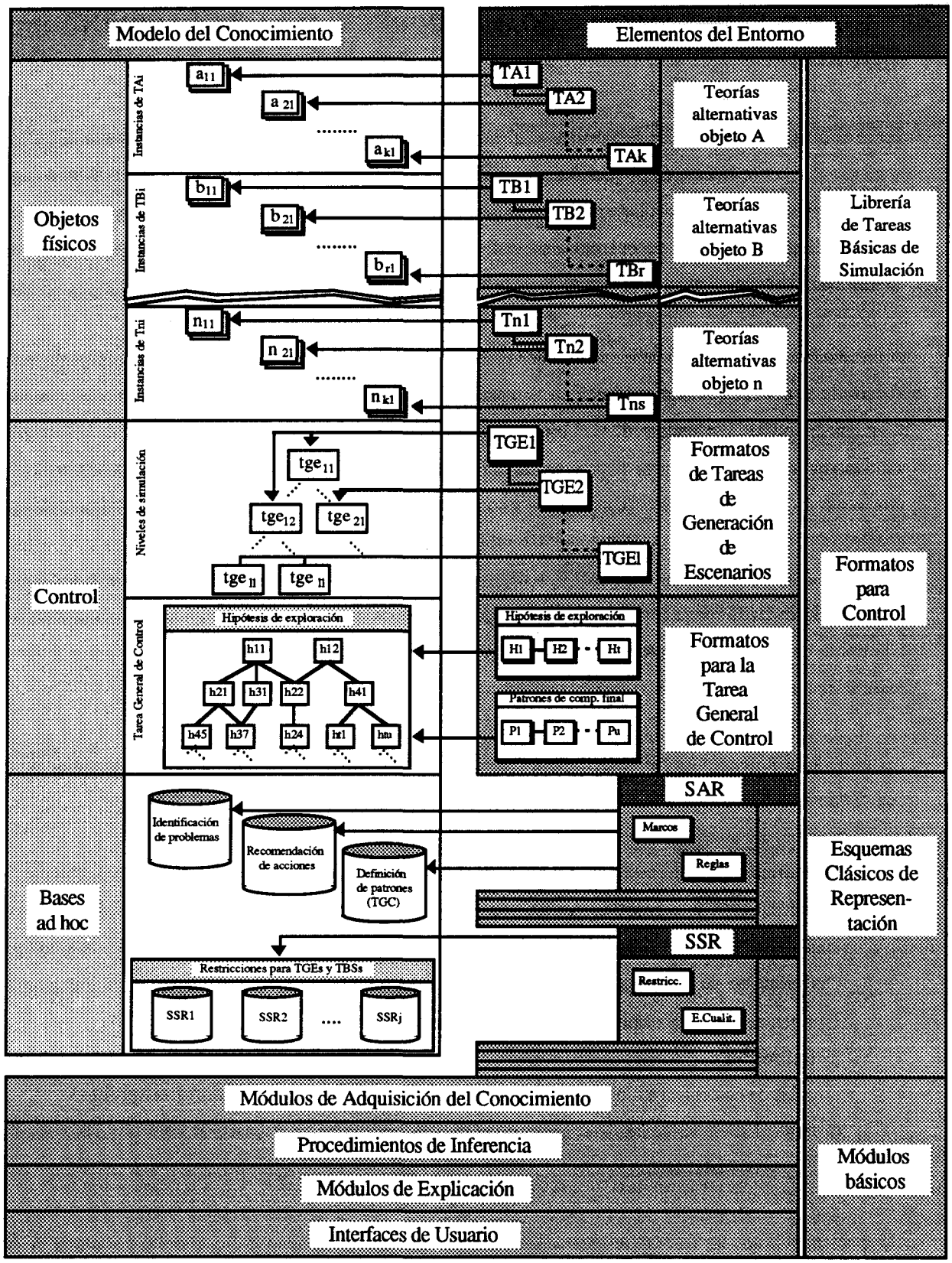


Figura 7a: Entorno y definición de un modelo.

Acorde con las ideas anteriores, un entorno construido alrededor de la arquitectura deberá suministrar objetos predefinidos (tipos de tareas) y herramientas software de soporte para la representación del conocimiento necesario, tanto para la simulación como para el control. La figura 7a muestra los elementos de un entorno así definido y refleja el proceso de construcción del modelo del conocimiento de un caso de estudio.

En los apartados siguientes se comentan los elementos del entorno desde el punto de vista de su utilización. En [Alonso et.al 92b] se encuentra una descripción más detallada del lenguaje DECON como guía de utilización.

7.1. ENTORNO CLASICO DE REPRESENTACION

El entorno clásico de representación (marcos, reglas y restricciones) se utiliza, principalmente, para la definición de las bases de conocimiento para inferencia de problemas así como las bases de evaluación de comportamientos e inferencia de patrones utilizadas por la tarea general de control. En la figura 7b se muestra la definición genérica, en el lenguaje DECON, de los conceptos de Hipótesis de Exploración y Patrón de Comportamiento Final detallados en el capítulo 6. Como se puede apreciar en la figura, ambos tipos de unidades cognitivas se definen como marcos clásicos de Minsky.

```
OBJETO: Patron de comportamiento final SUBCLASE DE Objeto
   VARIABLES: Interviene (Si, No) = Si [Constante];
              Activo (Si, No) = Si [Reglas];
              Verificado (Si, No) = No [Reglas];
              Valoracion Actual (INSTANCIAS DE Niveles] = Normalidad [Reglas];
              Valoracion Prevista [INSTANCIAS DE Niveles] = Normalidad [Reglas];
              Gradiente de cambio (INSTANCIAS DE Gradientes) = Nulo [Reglas];
              Superficie (NUMERICO) [Constante];
              Interes (NUMERICO) [F_Interes, ()];
              Area de influencia (INSTANCIAS DE TBS) [Constante]


OBJETO: Hipotesis de exploracion SUBCLASE DE Objeto.
   VARIABLES: Subhipotesis de (INSTANCIAS DE Hipotesisde exploracion] [Constante];
              Representantes (INSTANCIAS DE Patron de comportamiento final) [Constante];
              Nivel de activacion (Si, No, Posible) = Posible [F_Nivel de activacion, ()];
              Interes (NUMERICO) [F_Interes, ()];
              Area de exploracion (INSTANCIAS DE TBS) [Reglas]
```

Figura 7b: Definición de patrones e hipótesis.

En el ejemplo de la figura 7b, la definición realizada para *Patrón de comportamiento final* se interpreta de la forma siguiente:

- *Patrón de comportamiento final* es una clase definida como descendiente directo de la raíz de la jerarquía de objetos (el objeto *Objeto*).
- *Interviene* es una constante discreta que toma valores en el dominio (*Si, No*) y cuyo valor asignado es *Si*.
- *Activo* es una variable discreta que toma valores en el dominio (*Si, No*), se evalúa mediante reglas y su valor por defecto (si no se dispara ninguna regla) es *Si*.
- *Valoración Actual* es una variable discreta que toma valores en el dominio de los *Niveles* (objetos descendientes de la clase *Niveles*), se evalúa mediante reglas y su valor por defecto es *Normalidad*.
- Interés es una variable numérica que se evalúa mediante el método ad-hoc *F_Interes*, que acepta como parámetro al propio patrón definido. Para una instancia concreta de patrón, el diseñador puede redefinir la forma de evaluación.
- Etc.

En cada dominio concreto cabrá siempre la definición, que el lenguaje posibilita, de conceptos intermedios como subclase de los conceptos generales y que, de esta forma, pasarán a formar parte del entorno mismo. Por ejemplo, en el campo de la hidrología, cabe definir tipos más específicos de patrón dependiendo de los tipos de variables que se pretenden controlar. Así, se definirán patrones de lluvia, de caudal, de niveles en cauces, etc, que podrán añadir atributos nuevos sobre el concepto general e, incluso, redefinir algunos de los existentes.

Como ya se apuntó en el apartado 6.3 Sistemas Básicos de Representación e Inferencia, cada variable definida en un objeto DECON lleva asociada su método de obtención. Cabe incluir métodos ad-hoc programados por el diseñador o bien, caso más general, utilizar uno de los dos métodos distinguidos:

1) Evaluación mediante subbase de reglas.
2) Obtención de valor como parte de un proceso de satisfacción de restricciones

Las reglas DECON constituyen una implementación parcial del calculo de predicados. En general adoptan la forma, tanto en premisas como en conclusiones, de las clásicas ternas Objeto-Atributo-Valor con la sintaxis:

(*Objeto*) *Atributo* ES *Valor*

Eventualmente, sin embargo, es posible utilizar variables (a unificar durante el proceso de inferencia) ocupando plaza de objeto o de valor. En la figura 7b, por ejemplo, el Area de exploración de una Hipótesis de exploración podría definirse mediante una regla DECON como la unión de las áreas de influencia de sus patrones representantes:

PARA TODA ?h *Hipótesis de exploración*
SI (?h) *Representantes* ES ?p,
(?p) *Area de Influencia* ES ?Tbs
ENTONCES (?h) *Area de exploración* ES ?Tbs

Las reglas se utilizan, generalmente, para representar conocimiento de tipo deductivo pero admiten, además, representar conocimiento de tipo procedural, utilizando la técnica del paso de mensajes entre objetos. Por ejemplo, en la regla anterior el diseñador podría desear obtener una traza de las áreas de exploración de cada hipótesis, enviando un mensaje de escritura a consola:

PARA TODA ?h *Hipótesis de exploración*
SI (?h) *Representantes* ES ?p,
(?p) *Area de Influencia* ES ?Tbs
ENTONCES (?h) *Area de exploración* ES ?Tbs,
-> *Consola*, Escribe! {"La tarea $0 es parte del área de exploración de la
hipótesis $1", ?Tbs, ?h}

Por su parte, la utilización de restricciones será más indicada, en general, para representar conocimiento de control. Por ejemplo, para filtrado de escenarios de comportamiento (regresión de patrones de comportamiento final) en una tarea de generación de escenarios, para filtrado de un "envisionment" parcial obtenido en una tarea básica de simulación, etc. Por ejemplo, en el dominio de una planta química, podría plantearse la modelización de un reactor químico *Reactor ch32*[1]:

OBJETO *Reactor ch32* ES UN Objeto
CONSTANTES
Descripcion (ALFA) = "Reactor químico";
Sensores (INSTANCIAS DE Sensor) = *FIC 109, TI 100 12, TIC 107, TIC 114, FIC 108*
VARIABLES
dFlujo de salida (INSTANCIAS DE *Signo*) [*F_Evolución* ((*FIC 109*)*Tendencia*)];
dTemperatura de entrada (INSTANCIAS DE *Signo*) [*F_Evolución* ((*TI 100 12*)*Tendencia*)];
dTemperatura de reaccion (INSTANCIAS DE Signo) [*F_Evolución* ((*TIC 107*)*Tendencia*)];
dTemperatura de salida (INSTANCIAS DE Signo) [*F_Evolución* ((*TIC 114*)*Tendencia*)];
dFlujo de aire (INSTANCIAS DE *Signo*) [*F_Evolución* ((*FIC 108*)*Tendencia*)];
dFlujo de refrigerante (INSTANCIAS DE *Signo*);
dFlujo de evaporacion (INSTANCIAS DE *Signo*);

1 Ejemplo extraído de [Alonso et al. 92b].

dEntalpia (INSTANCIAS DE *Signo*);
dFuga (INSTANCIAS DE *Signo*);
dFuego (INSTANCIAS DE *Signo*);
dFuga en refrigeradores (INSTANCIAS DE *Signo*)

La letra *d*, al principio de cada variable, significa derivada, tomando valores dichas variables en el espacio cualitativo de los signos {-, 0, +}. Supónganse definidos, además, un conjunto de sensores estandar *FIC 109, TI 100 12, TIC 107*, etc. Finalmente, los métodos del tipo [*F_Evolución* ((*FIC 108*)*Tendencia*)] asignan los valores de partida a cada variable, como consecuencia de aplicar evolución temporal a una tendencia suministrada por el sensor correspondiente.

El modelo de comportamiento del reactor *Reactor ch32* se podría regir por la base en restricciones siguiente:

BASE DE RESTRICCIONES *Modelo de reacción*
ESPACIO CUALITATIVO *Signo*
USANDO *Reactor ch32*
RESTRICCIONES
dFlujo de salida :- dFuga en refrigeradores :- dFlujo de aire = dTemperatura de reaccion;
dFlujo de evaporacion :- dFuga = dFlujo de salida;
dEntalpia = dFlujo de evaporacion;
dTemperatura de entrada: - dTemperatura de salida: + dFuego externo = dEntalpia;
dPerdidas de frio :- dFlujo de refrigerante = dTemperatura de entrada;
dTemperatura de salida = dFlujo de refrigerante
FIN RESTRICCIONES

La acción de ambos tipos de métodos, Reglas y Restricciones, está debidamente coordinada en el seno del lenguaje DECON, de manera que es posible asignar valores a una variable mediante una base de reglas y después restringir esos valores como consecuencia de ejecutar un sistema de satisfacción de restricciones en el que se halle inmersa la variable. Esta situación es muy frecuente, por ejemplo, en las bases pertenecientes a las tareas de generación de escenarios.

7.2. LIBRERIA DE TAREAS BASICAS DE SIMULACION.

Una librería de tipos de Tareas Básicas de Simulación es el punto de partida para la modelización del comportamiento físico de los componentes involucrados, de forma que se tenga un "stock" de teorías alternativas sobre formas de razonar en cada tipo de objeto físico. Estas clases de tareas incluirán bases de conocimiento

definidas a nivel general mediante los correspondientes esquemas de representación. El constructor de un modelo, al definir su caso de estudio, creará instancias concretas de esos conceptos generales, heredando el conocimiento del tipo de tarea elegido, así como la linea general de razonamiento. Naturalmente, el lenguaje posibilita la definición de los detalles particulares necesarios en cada caso, bien por redefinición o por extensión del concepto general. El entorno debe suministrar herramientas de ayuda para ambos aspectos, actuando como guía del proceso de generación de instancias.

La figura 7c muestra la definición en DECON de una subclase de la clase general Tarea Básica de Simulación, que admite como entrada una o varias series temporales de datos y emite predicciones de respuesta en forma también de series y en un cierto horizonte temporal. La forma de razonamiento en este caso sería basado en transiciones explícitas, como ya se explicó en el capítulo 6.

```
TBS: Tarea basica de simulacion temporal SUBCLASE DE Tarea basica de simulacion
    ENTRADAS: Solicitacion (INSTANCIAS DE Serie temporal)
    PROPIEDADES: Tiempo inicial de simulacion (NUMERICO) [Constante];
                 Base de tiempos (NUMERICO) = 0.12 [Constante] # medido en horas
    VARIABLES INTERNAS:
        Tiempo inicial de solicitacion (NUMERICO) [F_Tiempo inicial, ()Solicitacion];
        Tiempo de simulacion (NUMERICO) [F_Incremento, ()Tiempo inicial de simulacion,
                                                        ()Base de tiempos]
    SALIDAS: Respuesta (INSTANCIAS DE Serie temporal)
    ESTADO:
    CREACION:
    INICIO:
    PROCESO:
    FINAL:
FIN TBS
```

Figura 7c: Clase de Tarea Básica de Simulación.

Notese como, al ser independiente de cualquier dominio, la definición anterior debe ser necesariamente muy simple, recogiendose únicamente como predefinidos los parámetros necesarios para el control de las campañas de simulación. Un entorno construido para una clase de dominio suministrará conceptos intermedios con mayor contenido. En el apéndice A se recogen definiciones de este tipo, en el campo de la hidrología, para las clases Tarea Básica de Simulación de Areas Receptoras.

7.3. FORMATO DE DEFINICION DE TAREAS DE GENERACION DE ESCENARIOS.

El entorno suministra una definición genérica de Tarea de Generación de Escenarios (ver figura 7d).

```
TGE: Formato general SUBCLASE DE Objeto
   VARIABLES:
     # variables intermedias para su uso en proceso de razonamiento de la TGE

   ESCENARIO:
     # formato (Objeto) Atributo [[EXCEPTO (Objeto) Atributo] (Objeto) Atributo]*
     # Tareas entrantes a la TGE con su variable de conexión.

   PERTINENCIA: # Linea de razonamiento de pertinencia (regresion de patrones)
    [[BASE DE REGLAS:  # definición de una base de reglas
       OBJETIVOS:
         # Formato (Objeto) Atributo [(Objeto)Atributo]*
         # variables a deducir por la base de reglas
      FIN BASE] |
     [BASE DE RESTRICCIONES: # definición de un sistema de restricciones.
       ESPACIO CUALITATIVO: # nombre del espacio cualitativo en el que opera el sistema
       RESTRICCIONES: # definición de restricciones del sistema
      FIN RESTRICCIONES]]*

   VEROSIMILITUD: # Linea de razonamiento de verosimilitud
    [[BASE DE REGLAS:  # definición de una base de reglas
       OBJETIVOS:
         # Formato (Objeto) Atributo [(Objeto)Atributo]*
         # variables a deducir por la base de reglas
      FIN BASE] |
     [BASE DE RESTRICCIONES: # definición de un sistema de restricciones.
       ESPACIO CUALITATIVO: # nombre del espacio cualitativo en el que opera el sistema
       RESTRICCIONES: # definición de restricciones del sistema
      FIN RESTRICCIONES]]*

   [PATRON REFERENCIA:
     # formato (Objeto) Atributo [[EXCEPTO (Objeto) Atributo] (Objeto) Atributo]*
     # Variables para actuar como patrón de referencia en generación de escenario]

FIN TGE
```

Figura 7d: Formato de definición de una Tarea de Generación de Escenarios.

En este caso, el usuario debe especificar:

1) Variables intermedias (las que se necesiten) para razonamiento en la TGE.
2) Las variables de definición del escenario entrante. Serán variables de salida de tareas antecedentes a la TGE en el proceso de simulación. Se pueden especificar individualmente o agrupadas por clases de tareas básicas de simulación (por ejemplo, un escenario entrante podría definirse como (*Area receptora*) *Caudal* EXCEPTO (*Barranco de la casella*) *Caudal*, significando el caudal de salida de cada área receptora de lluvia en una cuenca hidrográfica a excepción del barranco de la Casella).

3) Bases de conocimiento en reglas y/o restricciones para realizar la regresión de patrones de comportamiento final (filtrado por pertinencia según objetivos).
4) Bases de conocimiento en reglas y/o restricciones para aplicar los criterios de verosimilitud.
5) Opcionalmente, variables para actuar como patrón de referencia en la generación de escenarios, con el mismo formato que las variables de definición del escenario entrante.

Nuevamente, la clase general de TGE recoge únicamente la estructura pero vacía de contenido al ser independiente de cualquier dominio. En el apéndice A se recoge un ejemplo completo de definición sobre DECON, en el campo hidrológico, para la TGE Meteorológica (tarea de generación de escenarios de respuesta en Zonas de Lluvia).

7.4. MODULOS BASICOS.

Finalmente, el cierre del entorno requiere un conjunto de procedimientos básicos de apoyo a la definición y operación de modelos:

1) Un procedimiento de adquisición de conocimiento para guiar al usuario en la creación de un modelo. Cada modelo se producirá como una instancia de la gramática del lenguaje DECON (o bien de extensiones o particularizaciones de este), almacenada en ficheros de texto para su posterior compilación. El proceso de compilación leerá el modelo de esos ficheros y:

   - Generará la estructura del conocimiento en memoria para inferencia.
   - Rechazará el modelo si se encuentran errores.

2) Un motor de inferencia para aplicar la linea general de razonamiento (es decir, la estrategia general), utilizando la estructura de conocimiento compilada. Aunque la arquitectura suministra la forma general de razonamiento descrita en el apartado 5.2.6, lo normal será que esta requiera adaptaciones cuando se trate de aplicar a diferentes clases de dominios.

3) Módulos de explicación generales, para los paradigmas clásicos de representación basados en reglas y restricciones, y particularizados para cada tipo de tarea en el caso de las TBSs y las TGEs.

7) Finalmente, una interfaz de usuario. Esta interfaz será necesariamente distinta para cada posible entorno, con objeto de adaptarse a las peculiaridades de cada clase de dominio. En el anejo A se recogen esquemas de interfaz para el dominio hidrológico.

7.5 EL NUEVO ESCENARIO DE LA INGENIERIA DEL CONOCIMIENTO.

Las fases de construcción, y las herramientas enumeradas, no pretenden conformar una metodología en el sentido riguroso del término. De hecho, la definición de la arquitectura y los entornos especializados que sobre ella se construyan deben aportar, como efecto lateral, una disminución de las necesidades metodológicas al realizar el proceso de adquisición del conocimiento.

En efecto, de todo lo anterior se deduce fácilmente que la tarea de la ingeniería del conocimiento, para la construcción de modelos sobre este tipo de arquitecturas y entornos, no se basará en el clásico ciclo de entrevistas ingeniero-experto(s) para, a partir de las cuales, detallar los modelos mediante elementos básicos de representación. Por el contrario, surge un nivel intermedio de definición de la maquina cognitiva aglutinante de los conceptos a los que referir la forma de entender y resolver problemas de una cierta clase, y capaz de comprender los principios básicos sobre los que aquellos descansan [Cuena 91].

Este concepto de la ingeniería se ilustra en la figura 7e[2], en la que aparecen a un nivel básico las diversas técnicas de representación y razonamiento y, a un nivel intermedio, las arquitecturas de posibles máquinas cognitivas, todo ello en oposición a la linea clásica (mucho más basada en los aspectos metodológicos) apoyada en arquitecturas generales.

De acuerdo con las consideraciones anteriores, el nuevo escenario de la ingeniería del conocimiento tiene una doble vertiente:

1) Creación de modelos de abstracción a nivel cognitivo del universo de conceptos.
2) Diseño de representaciones simbólicas computables capaces de procesar los modelos que se construyan sobre la arquitectura cognitiva.

La identificación del nivel intermedio de maquinas cognitivas especializadas en clases de problemas (en general áreas profesionales) supone una simplificación del proceso de adquisición del conocimiento, debido a dos consideraciones:

1) Sugiere de entrada una estructuración y organización del conocimiento. Esto debe implicar procesos más sistemáticos de adquisición.
2) La máquina cognitiva llevará ya incorporado conocimiento de tipo general sobre la clase de problemas en que se especializa la arquitectura, lo cual implica que el experto, en cada caso, solo tiene que aportar el conocimiento específico.

---

2 Figura adaptada de [Cuena 91].

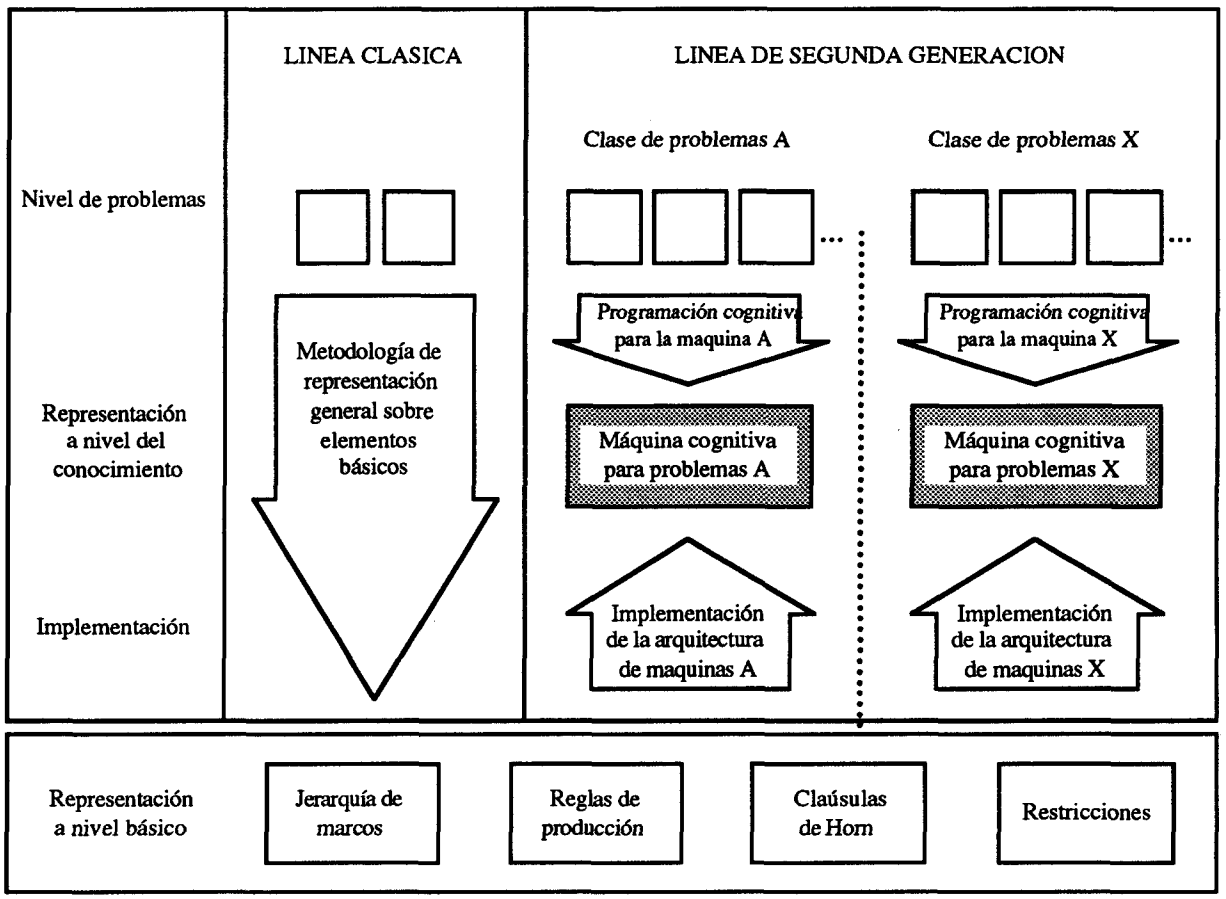


Figura 7e: Los enfoques de la ingeniería del conocimiento.

En resumen, cabe afirmar que este tipo de arquitecturas de segunda generación, si bien más complejas de diseñar e implementar que una arquitectura de primera generación, requieren a cambio métodos de adquisición del conocimiento más simples que serán, en consecuencia, más fiables.

8. CONCLUSIONES.

8.1. APORTACIONES DE LA TESIS.

La principal aportación de la tesis ha sido la concepción y diseño de un tipo de arquitecturas de representación del conocimiento en el entorno de los sistemas físicos, con las siguientes características generales:

1) Incluye no solo las formas clásicas de representación, sino también los paradigmas de razonamiento, más especializados, de uso común por los profesionales en ingeniería. Para ello, se definen las Tareas Básicas de Simulación (TBS) como vehículo para representar conocimiento sobre el comportamiento de sistemas, mediante técnicas de simulación cualitativa.
2) Representa explícitamente el conocimiento de control a todos los niveles. Para ello se definen dos tipos de tareas: la Tarea General de Control (TGC), para gobierno de la estrategia general, y las Tareas de Generación de Escenarios (TGE) para filtrado de las evoluciones posibles de un sistema en fase de razonamiento profundo. Ambos tipos de tareas funcionan de forma cooperante, de manera que las TGEs se apoyan, para efectuar el filtrado de comportamientos, en Patrones de Comportamiento Final deducidos previamente por la TGC y cuya regresión permite, de hecho, llevar a cabo procesos de simulación guiada por objetivos. Esta forma de control es novedosa en sistemas de simulación cualitativa.
3) Aporta un nuevo lenguaje para definición del conocimiento (lenguaje DECON), como soporte de este tipo de arquitecturas.

La clase de arquitecturas presentada muestra la viabilidad de construcción de una clase de sistemas donde se aunan las técnicas de la Inteligencia Artificial con la tradición modelística constituyendo, en consecuencia, una vía de integración que puede ser de gran utilidad en el campo de la ingeniería, donde existe un importante "stock" de procedimientos instrumentados como base para este tipo de aplicaciones.

La tesis aporta un entorno específico en el dominio hidrológico (el SIRAH) como ejemplo de (o como un paso hacia) una posible nueva generación de herramientas de Inteligencia Artificial, alejadas del concepto clásico de entorno de software para construcción de sistemas expertos, pero más útiles que estos para resolver problemas complejos en ingeniería.

Esta clase de nuevas herramientas son necesarias y útiles. La tesis pone de manifiesto como los paradigmas uniformes basados en marcos, reglas, etc, no pueden proporcionar la estructura requerida para ser lo suficientemente fieles al conocimiento que los profesionales poseen y utilizan sobre su forma de entender la resolución de problemas en las diferentes ramas de la ingeniería.

De acuerdo con estos conceptos de arquitecturas y entornos, la tesis contribuye a alcanzar una forma de programación más cognitiva; es decir, más basada en las formas de entender y no tanto de computar. En base a ello, se propone el escenario para realizar la ingeniería del conocimiento, en el cual aparece un nivel intermedio de maquina cognitiva, especializada en la rama profesional correspondiente y capaz de "entender" sobre las teorías básicas en ese dominio. Esto debe permitir la adquisición del conocimiento profesional basado en los modelos tradicionales existentes en la práctica de la ingeniería ya que la mayor parte del razonamiento de sentido común de los profesionales humanos se realiza basándose en datos y resultados de librerías de modelos, sin entrar en el conocimiento contenido implícitamente en los mismos, que se asume como bien conocido. El proceso de adquisición se efectúa, de esta forma, articulado por la propia arquitectura, facilitando la labor -a la par que disminuyendo el papel, del ingeniero del conocimiento. El tipo de entornos, con aporte cognitivo especializado, propuestos por la tesis, tienen capacidad para representar estrategias generales de resolución de problemas en un determinado ámbito pero, además, posibilitan la definición de módulos tarea que, incorporando modelos cualitativos, sean capaces de "comprender" las teorías básicas.

Las consideraciones anteriores trabajan en la linea de permitir al profesional acercarse a la tarea de construir un modelo complejo dialogando con el entorno en su lenguaje. La experiencia acuñada en el Laboratorio de Sistemas Inteligentes en la construcción de modelos hidrológicos complejos por parte de expertos en ese área, utilizando el entorno SIRAH, así lo confirma.

Finalmente, todas estas ideas han cristalizado en la que, quizá, es la contribución más tangible de la tesis: la definición de un lenguaje formal de programación cognitiva (DECON) como soporte a la arquitectura. El lenguaje DECON permite programar, en sintaxis pseudo-castellana, unidades cognitivas de cada uno de los tipos considerados, así como el conocimiento para establecer las estrategias generales.

El lenguaje DECON puede encuadrarse en la linea de los lenguajes capaces de integrar diferentes paradigmas de programación de forma cooperante, como es el caso de LOOPS [Stefik, Bobrow 83]; es decir:

1) Programación orientada a procedimientos (programación imperativa).
2) Programación orientada a objetos.
3) Programación basada en reglas.
4) Programación orientada a datos (valores activos).

El lenguaje DECON incluye, en terminos generales, similares capacidades básicas que LOOPS (ha de reconocerse más débil que este respecto de los mecanismos de herencia) pero con mayor potencial ya que:

1) Admite también programación basada en restricciones como paradigma básico.

2) Permite realizar en él, directamente, programación cognitiva en base a tareas.

DECON auna, además, características de los lenguajes para definición de tareas como CSRL (definición de tareas de tipo clasificativo [Chandrasekaran 86, 87]) y DSPL (definición de tareas de diseño [Brown, Chandrasekaran 89]). Es posible emular a ambos en DECON con la misma capacidad de representación. En el Laboratorio de Sistemas Inteligentes se han realizado experimentos en este sentido, programando ambos tipo de tareas, en DECON, para diagnóstico y reparación en el dominio de los sistemas mecánicos. El lenguaje extiende esas capacidades al posibilitar la definición de tareas para razonamiento cualitativo sobre el comportamiento de sistemas físicos, así como tareas específicas para efectuar control guiado por objetivos.

El lector puede encontrar extraño el hecho de que la tesis no incluya un capítulo específico para describir exhaustivamente las capacidades del lenguaje DECON. No se ha considerado necesario ya que el lenguaje es un fiel reflejo de la arquitectura y, por otra parte, el anejo B recoge su definición formal completa. No existe en el estado actual del arte un lenguaje que conjugue su potencial y características.

8.2. LIMITES DE LA TESIS: LINEAS FUTURAS DE INVESTIGACION.

Aunque el proyecto investigador puede considerarse, en cuanto a base para una tesis doctoral, finalizado (aceptable satisfacción de los objetivos marcados y existencia de contribuciones reales al estado del arte) conviene concluir con un obligado ejercicio de modestia ya que, como corresponde a una labor investigadora, siempre será mayor el trabajo pendiente que el realizado.

Las posibilidades de continuación del trabajo investigador y de desarrollo emanan de las limitaciones de la tesis, que podrían enumerarse como sigue:

1) El lenguaje DECON goza de las características positivas antedescritas, pero no está exento de aspectos criticables. En primer lugar es, en algunos casos, poco intuitivo a la hora de codificar conocimiento y, de igual forma, poco uniforme en su definición. Su concepción modular (ventaja innegable) y el diseño independiente de las gramáticas que lo componen han dado lugar a esa falta de uniformidad. El lenguaje es fácil de aprender, pero impone limitaciones (algunas derivadas de su sintaxis, otras derivadas de su propia concepción) difíciles de comprender, en ocasiones, por el profesional que lo maneja. En la linea de facilitar la utilización del lenguaje sería concebible, por ejemplo, construir interfaces inteligentes, capaces de comprender la sintaxis y acciones semánticas asociadas al lenguaje. Esto podría conseguirse a traves de ventanas DECON que guiaran al profesional en el proceso de configuración de un modelo, sugiriendo prototipos de definición de objetos, recuperando errores de forma interactiva, etc.

En segundo lugar, el lenguaje dista mucho de ser un producto terminado. Es un producto de laboratorio, y así debe entenderse. No admite comparación, en este sentido, con los lenguajes mencionados sobre los que se ha basado su diseño y a los que supera en capacidad para hacer programación cognitiva, pero lejos de ellos en aspectos de soporte en la utilización del lenguaje, en particular -como ya se ha señalado, en lo relativo a ayudas interactivas a la tarea de modelizar.

2) La arquitectura es demasiado rígida todavía, en especial al recoger conocimiento de control. Si bien supera las limitaciones, ya comentadas, de las arquitecturas de Chandrasekaran en ese sentido, no es aún general en el grado deseable. Por ejemplo, la arquitectura impone las posibles topologías de los modelos que se construyen. Aunque está demostrada su aplicabilidad a dominios diferentes del que sirvió como núcleo de investigación (como lo prueba el hecho de que se haya utilizado en campos tan distintos como la predicción en el dominio hidrológico y el diagnóstico profundo en instalaciones de la industria química), esa migración requiere adaptaciones a veces costosas, si no a nivel cognitivo, si a nivel simbólico y de implementación e instrumentación.
3) No se ha alcanzado todavía el objetivo de programación cognitiva plena, quizá por la limitación anterior, requiriendose en exceso soporte informático ad hoc. Aunque la tesis propugna, de hecho, una linea de entornos específicos por ramas profesionales, una vez en un dominio determinado el papel del software es aún demasiado relevante.
4) La arquitectura posibilita la inclusión de funciones de aprendizaje empírico (es decir, no basadas en explicaciones sino en muestras significativas de comportamiento). En especial, la concepción y diseño de las Tareas Básicas de Simulación se presta con facilidad a su superficialización. Para un TBS podrían generarse, por ejemplo, reglas de inferencia que resuman su funcionalidad, a partir de muestras significativas de comportamiento susceptibles de ser tratadas con algún método automático de aprendizaje tipo [Quinlan 79] o [Michalski et al. 86]. Ha de reconocerse, sin embargo, que este aspecto no se ha abordado más que, escasamente, en el plano teórico.
5) En el plano de la implementación, cabe afirmar que la eficiencia computacional es mejorable. Si bien por vía de los instrumentos de desarrollo utilizados el rendimiento obtenido es aceptable, este es un aspecto al que todavía no se ha prestado la atención necesaria. Por ejemplo, la ejecución de los procesos de inferencia admite paralelismo en, al menos, dos casos:
    - Exploración de hipótesis entregadas por la tarea general de control.
    - Ejecución de tareas de predicción de comportamiento en un mismo nivel de simulación, estableciendo como punto de sincronización la tarea de generación de escenarios correspondiente a dicho nivel.
6) Finalmente, el lector habrá notado como la tesis no describe las capacidades explicativas de la clase de arquitecturas propuesta. Ello es debido a que, si bien la arquitectura diseñada contiene módulos explicativos, estos son los normales ya conocidos y aplicables a las formas clásicas de representación (marcos, reglas, etc). Aunque concebidos teóricamente, no se han desarrollado mecanismos de

mecanismos de explicación adecuados para justificar el razonamiento profundo (tareas de razonamiento cualitativo) y los definidos para el razonamiento de control (TGEs) son relativamente simples. No era aconsejable, por tanto, incluirlos en esta memoria.

Las limitaciones enumeradas no deben ser sino acicate para futuros trabajos. Sirvan, al mismo tiempo, como cauce para un apunte final: la linea de investigación emprendida por el Laboratorio de Sistemas Inteligentes, en arquitecturas de segunda generación, ha sido tan sugerente como prometedora. Cabe desear, por consiguiente, que proyectos como el que ha servido de marco al trabajo realizado tengan su lógica continuación. Ello permitiría incidir en esta labor que, si bien puede considerarse de investigación básica, posee el innegable valor añadido de ofrecer resultados prácticos a corto y medio plazo.

Bibliografía.

[Abbott 83] Abbott R.: "Program Design by Informal English Descriptions". ACM, Vol. 26, No. 11, pag 882-894, November 1983.

[Alonso et al. 89] Alonso M., Cuena J., Molina M.: "CYRAH: Un Ejercicio de Integración de Modelos de Simulación y Reglas para Representación del Comportamiento Físico". en las Actas de la III Reunión Técnica de la Asociación Española para la Inteligencia Artificial, Madrid, 1989.

[Alonso et al. 90a] Alonso M., Cuena,J., Molina,M.: "SIRAH: Una arquitectura para conocimiento profesional", II Congreso Iberoamericano de Inteligencia Artificial, México, 1990.

[Alonso et al. 90b] Alonso M., Cuena,J., Molina,M.: "Sirah: An architecture for a professional intelligence" en Proceedings of the European Conference on Artificial Intelligence, Estocolmo, Suecia, 8-1990.

[Alonso et al. 92a] Alonso M., Cuena J., Reig B.: "SIRAH: A Software Environment for Advanced Knoweledge-Based Models for Flood Management". HYDROSOFT-92, Universidad Politécnica, Valencia, 1992.

[Alonso et al. 92b] Alonso M., Izquierdo E., Salmerón A.: "DECON: Guía de Utilización. Versión preliminar". Documento interno. EQ Sistemas Inteligentes S.L., Diciembre 1992.

[Alonso, Cuena 89] Alonso,M., Cuena,J.: "El medio ambiente de la representación del conocimiento Sarah: Versión preliminar" en Civil Engineering Expert Systems, Civil Engineering European Cursus, Madrid, 6-2-1989.

[Baltzer et al. 80] Balzer R., Erman L.D., London R. y Williams C.: "HEARSAY-III: A domain-independent framework for expert systems". en Proc. AAAI-80, Stanford, CA, 1980.

[Bonissone, Valavani 85] Bonissone,P.P., Valavani,K.P.: "A comparative study of different approaches to qualitative phisics theories" en IEEE, pag. 236, 1985.

[Booch 83] Booch,G.: "Software engineering with Ada", Companion Series-Benjamin Cummings, 1983.

[Booch 86] Booch,G.: "Object-oriented development" en IEEE Transactions on Software Engineering, Vol.SE-12, 2-1986.

[Brachman, Smith 80] Brachman, R.J. and Smith, B.C.: "Special issue on Knowledge Representation". SIGART Newsletter 70, pags 1-38, 1980.

[Bredeweg, Wielinga 88] Bredeweg,B., Wielinga,B.: "Integrativy qualitative reasoning approach" en Proceedings of the European Conference on Artificial Intelligence, ECAI-88, Munich, Alemania, 1988.

[Breuker, Wielinga 84a] Breuker B.J., Wielinga B.J.: "Interpretation of verbal data for Knowledge Adquisition". Proc. ECAI 84, North-Holland, Amsterdan, pag. 41-50, 1984.

[Breuker, Wielinga 84b] Breuker,J., Wielinga,B.: "Tecniques for Knowledge Elicitation Analysis", Report 1.5 Esprit Project 12, 1984.

[Breuker, Wielinga 85] Breuker,J., Wielinga,B.: "KADS: Structured Knowledge Acquisition for Expert Systems" en 5th International Workshop on Expert Systems and their Applications, pags. 887-900, Agence de L'Informatique, Avignon, Francia, 1985.

[Breuker, Wielinga 86] Breuker,J., Wielinga,B.: "Use of Models in Interpreting Verbal Data", Memorandum 93 of the Research Project "The Acquisition of Expertise", 1986.

[Brown, Chandrasekaran 83] Brown,D.C., Chadrasekaran,B.: "An approach to expert systems for mechanical design" en Proceedings of Trends and Applications on Automating Intelligent Behavior: Applications and Frontiers, pags. 173-180, 1983.

[Brown, Chandrasekaran 84] Brown,D.C., Chadrasekaran, B.: "Expert systems for a class of mechanical design activity" en Proceedings IFIP WG 5.2 Working Conference on Knowledge Engineering in Computer Aided Design, IEEE Computer Society, Budapest, Hungría, 1984.

[Brown, Chandrasekaran 89] Brown,D.C., Chandrasekaran,B.: "Design problem solving" en Knowledge structures and control strategies, Morgan Kaufman Publishers, San Mateo, CA, USA, 1989.

[Buchanan et al. 84] Buchanan B.G., Shortlife E.H.: "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project"., Addison-Wesley, Reading, MA, 1984.

[Bylander, Chandrasekaran 87] Bylander,T., Chandrasekaran,B.: "Generic tasks in knowledge-based reasoning: The right level of abstraction for knowledge acquisition" en International Journal of Man-Machine Studies, Vol.26, No.2, 1987.

[Clancey 83] Clancey,W.J.: "The advantages of abstract control knowledge in expert systems design" en Proceedings of the National Conference on Artificial Intelligence, AAAI-83, Morgan Kaufmann, Washington DC, USA, 1983.

[Clancey 85] Clancey,W.J.: "Heuristic Classification" en Artificial Intelligence, Vol.27, pags. 289-350, Elsevier Science Publishers B.V., North-Holland, 1985.

[Constantine, Yourdon 87] Constantine L., Yourdon E.: "Structured Design: A Discipline of Computer Programs and Systems Design". Yourdon Press, 1987.

[Corkill et al. 82] Corkill D.D., Lesser V. y Hudlicka E.: "A goal-directed Hearsay-II architecture: Unifying data-directed and goal-directed control" en proc. National Conference on Artificial Intelligence, Pittsburg, PA, 1982.

[Corkill et al. 88] Corkill D.D., Gallagher K.Q. y Johnson P.M.: "Achieving Flexibility, Efficiency and Generality in Blackboard Architectures". Tech. rep. Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, 1988.

[Cox 85] Cox B.: "Software ICs and Objetive-C". Unix World, Spring, 1985.

[Cox 86] Cox B.: "Object Oriented Programming: An Evolutionary Approach". Addison Wesley, 1986.

[Cuena 83] Cuena J.: "The Use of Simulations Models and Human Advice to Build an Expert System for the Defense and Control of River Floods". (IJCAI-83), Karlsruhe, Kaufmann, 1983.

[Cuena 88a] Cuena J.: "The Qualitative Modeling of Axis-Based Flow Systems: Methodology and Examples". European Conference on Artificial Intelligence. Munich, Pitman, 1988.

[Cuena 88b] Cuena,J.: "Building expert systems based on simulation models: An essay in methodology" en Expert Systems Applications, Bole,L. y Coombs,M.J. (eds), Springer Verlag, 1988.

[Cuena 89a] Cuena,J.: "AURA: A second generation expert system for traffic control in urban motorways" en Proceedings of the 9th International Workshop on Expert Systems, EC2, Avignon, Francia, 1989.

[Cuena 89b] Cuena,J.: "Knowledge-based systems for aid in decision-making: Methodology and examples" en Perspectives in Artificial Intelligence, Vol.1, pags. 73-92., Campbell,J.A. y Cuena,J. (eds), Ellis Horwood, 1989.

[Cuena 89c] Cuena J.: "Sistemas Basados en el Conocimiento: Situación actual y Perspectivas". Civil Engineering Expert Systems. COMMET, Madrid, 1989.

[Cuena 91] Cuena,J.: "Arquitecturas de segunda generación", Lección en la Academia de Ciencias, Consejo Superior de Investigaciones Científicas, Madrid, 1991.

[Cuena et al. 91] CuenaJ., Molina M., Garrote L.: "An Architecture for Cooperation of Knowledge Bases and Quantitative Models: The CYRAH Environment". XI International Workshop on Expert Systems. Special conference on Second Generation Expert Systems. Avignon '91. EC2, 1991.

[Cuena, Campbell 89] Cuena,J., Campbell,J.A.: "Perspectives in artificial intelligence, Vol.I. Expert systems: Applications and Technical Foundations", Ellis Horwood series, 1989.

[Cuena, Salmerón 89] Cuena,J., Salmerón,A.: "Reasoning over the behavior of physical systems" en Civil Engineering European Cursus, Civil Engineering Expert Systems, artículo completado y revisado por Alonso M., Madrid, 1989.

[Cuena, Salmerón 91] Cuena,J., Salmerón,A.: "Arquitecturas de segunda generación para el diagnóstico profundo en instalaciones industriales" en las Actas de la IV Reunión Técnica de la Asociación Española para la Inteligencia Arificial, AEPIA-91, Madrid, 23-10-1991.

[Chandrasekaran 83] Chandrasekaran,B.: "Towards a taxonomy of problem solving types" en A.I. Magazine, Vol.4, No.1, pags. 9-17, 1983.

[Chandrasekaran 86] Chandrasekaran,B.: "Generic tasks in knowledge based reasoning: High level building blocks for expert systems design" en IEEE Expert, 1986.

[Chandrasekaran 87] Chandrasekaran,B.: "Towards a functional architecture for intelligence based on generic information processing tasks" en Proceedings of the 10th International Joint Conference on Artificial Intelligence, IJCAI-87, pags. 1183-1192, Morgan Kaufmann Publishers, Milán, Italia, 1987.

[Chandrasekaran et al. 91] Chandrasekaran B., Bhatnagar R., Sharma D.D.: "Real-Time Disturbance Control". Comunications of the ACM, Vol. 34, No.8, pags 33-47, August 1991.

[Chandrasekaran et al. 92] Chandrasekaran B., Johnson R., Smith W.: "Task-Structure Analysis for Knowledge Modeling". Comunications of the ACM, Vol. 35, No.9, pags 124-136, September 1992.

[Chandrasekaran, Milne 85] Chandrasekaran,B., Milne,R.: "Special section on reasoning about structure, behavior, and function" en Sigart Newletter, No.93, pags. 5-54, ACM, 1985.

[Chandrasekaran, Mittal 82] Chandrasekaran,B., Mittal,S.: "Deep versus compiled knowledge approaches to diagnostic problem-solving" en Proceedings of the National Conference on Artificial Intelligence, AAAI-82, Pittsburgh, PA, USA, 1982.

[Chandrasekeran, Goel 88] Chandrasekeran,B., Goel,A.: "From numbers to knowledge structures: AI perspectives on the classification task", LAIR Technical Report Library 217 CAE, Dept. of Computer Science, 2036 Nel Avenue Mall Columbus Ohio 43210-1277, The Ohio State University, Columbus, Ohio, USA, 1988.

[Ching-Chih, Chia-Hoang 88] Ching-Chih, H., Chia-Hoang, L.: "Comments on Mohr and Henderson's Path Consistency Algorithm" Artificial Intelligence nº 36 Pg: 125-130. 1988

[Chow et al. 87] Chow V., Maidment D., Mays L.: "Applied Hydrology". McGraw Hill, (Chapters 7-9), 1987.

[De Kleer, Brown 84a] De Kleer,J., Brown,J.S.: "Qualitative reasoning about physical systems" en Artificial Intelligence, Elsevier Science Publishers B.V., North-Holland, 1984.

[De Kleer, Brown 84b] De Kleer,J., Brown,J.S.: "A qualitative physics based on confluences" en Artificial Intelligence 24, pags. 7-83, Elsevier Science Publishers B.V., North-Holland, 1984.

[Dechter, Pearl 88] Dechter, R., Pearl J.: "Network-Based Heuristics for Constraint-Satisfaction Problems" Artificial Intelligence nº 34 Pg.: 1-38. 1988.

[Doyle 79] Doyle,J.: "A truth maintenance system" en Artificial Intelligence, Vol.12, pags. 231-272, Elsevier Science Publishers B.V., North-Holland, 1984.

[Doyle 82] Doyle,J.: "A glimpse of truth maintenance" en Artificial Intelligence: An MIT Perspective, The MIT Press, pags. 119-135, Cambridge, MA, USA, 1982.

[Erman et al. 81] Erman L., London P., Fickas S.: "The design and example use of Hearsay-III" en proc. 7th International Joint Conference on Artificial Intelligence, pag 409-415, August 1981.

[Erman, et al. 80] Erman,L.D., Hayes-Roth,F., Lesser,V.R., Reddy,D.R.: "The Hearsay II speech understanding system integrating knowledge to resolve uncertainty" en ACM Computing Surveys, Vol. 12 (2), pags. 213-253, 1980.

[Ernst, Newell 69] Ernst G., Newell A.: "GPS: A Case Study in Generality and Problem Solving". Academic Press, New York, 1969.

[Feigenbaum et al. 71] Feigenbaum E.A., Buchanan B.G. and Lederberg J.: "On Generality and Problem Solving: A Case Study Using the DENDRAL program". Machine Intelligence, American Elsevier, New york, pag 165-190, 1971.

[Fikes, Nilsson 71] Fikes,R.E., Nilsson,N.J.: "STRIPS: A new approach to the application of theorem proving to problem solving" en Artificial Intelligence, Vol. 2 (3-4), pags. 189-208, Elsevier Science Publishers B.V., North-Holland, 1971.

[Fontán 91] Fontán S.:"Integración de Simulación Cualitativa y Razonamiento Simbólico. Aplicación a un Sistema Experto en comportamiento de Cuencas Hidrográficas". Proyecto Fin de Carrera, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, 1991.

[Forbus 81] Forbus,K.D.: "Qualitative reasoning about physical processes" en Proceedings of the Seventh International Joint Conference on Artificial Intelligence, William Kaufman, Los Altos, CA, USA, 1981.

[Forbus 84] Forbus,K.D.: "Qualitative process theory" en Artificial Intelligence Vol.24, pags. 85-108, Elsevier Science Publishers B.V., North-Holland, 1984.

[Forbus 88] Forbus,K.D.: "Qualitative physics: past, present and future" en Exploring Artificial Intelligence: Survey talks from the National Conferences on Artificial Intelligence, Shrobe,H.E. (eds), Morgan Kaufmann Publishers, San Mateo, CA, USA, 1987.

[Forbus 89] Forbus,K.D.: "The qualitative process engine" en Readings in Qualitative Reasoning About Physical Systems, Weld,D.S. y De Kleer,J. (eds), Morgan Kaufmann, 1989.

[Forgy 82] Forgy,L.: "Rete: A fast algorithm for the many patterns / many objects match problem" en Artificial Intelligence, Vol.19, Elvesier Science Publishers B.V., North-Holland, 1982.

[Forgy, McDermott 77] Forgy C.L., McDermott J.: "OPS, a domain-independent production system languaje". en Proc. IJCAI-77, Cambridge MA, 1977.

[Freuder 78] Freuder, E. C.: "Synthesizing Constraint Expressions" Communications of ACM vol. 21, nº 11 Pg: 958-966. 1978.

[Freuder 82] Freuder, E. C.: "A Sufficient Condition for Backtrack-Free Search" Journal of the ACM vol. 29, nº 1 (1982) Pg.: 24-32.

[Garrote 90] Garrote,L.: "Modelos hidrológicos de ayuda a la decisión en tiempo real basados en técnicas de inteligencia artificial", Tesis doctoral, Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos, Universidad Politécnica de Madrid, Madrid, 1990.

[Goldberg et al. 83] Goldberg A., Robson D.: "Smalltalk-80: The Languaje and its Implementation". Addison-Wesley, 1983.

[Haralick, Elliott 80] Haralick R. M., Elliott G. L.: "Increasing Search Efficiency for Constraint Satisfaction Problems" Artificial Intelligence nº 14 Pg.: 263-313. 1980.

[Hayes 77] Hayes P.J.: "The logic of frames" en "Frame conceptions and text underestanding". Editado por D. Metzing. Berlin: Walter de Gruyter Co., pg 46-61, 1979.

[Hayes 79] Hayes P.J.: "The naive physics manifesto" en Expert Systems in the Microelectronic Age, Michie,D. (ed), Edimburgh University Press, Edinburgo, Gran Bretaña, 1979.

[Hayes 85] Hayes,P.J.: "The second naive physics manifesto" en Formal Theories of the Commonsense World, Hobbs,J.R. y Moore,R.C. (eds), Ablex, Gran Bretaña, 1985.

[Hayes-Roth 85] Hayes-Roth B.: "An Blackboard Architecture for Control". Artificial Intelligence, 26(3), pags. 251-321, Junio 1985.

[Hitchman et al. 89] Hickman F., Killin J., Land L., Mulhall T., Porter D., Taylor R.: "Analysis For Knowledge-Based Systems: a practical guide to the KADS methodology". Ellis Horwood Books in Information Technology, 1989.

[Izquierdo 92] Izquierdo E.: "Implementación de un sistema de satisfación de restricciones cualitativas". Proyecto de fin de carrera. Facultad de Informática de Madrid, 1992.

[Kernighan, Pike 84] Kernighan,B.W., Pike,R.: "The Unix programming environment", Prentice-Hall, inc, New Jersey, NJ, USA, 1984.

[Kolodner 87] Kolodner J.L.: "Extending Problem Solving Capabilities Through Case-Based Inference". Procc Fourth International Machine Learning Workshop. 1987.

[Kuipers 84] Kuipers,B.J.: "Commonsense reasoning about causality: Deriving behavior from structure" en Artificial Intelligence, Vol.24, pags. 169-203, Elvesier Science Publishers B.V., North-Holland, 1984.

[Kuipers 85] Kuipers,B.J.: "The limits of qualitative simulation" en Proceedings of the 9th International Joint Conference on Artificial Intelligence, IJCAI-85, William Kaufman Publishers, Los Angeles, CA, USA, 1985.

[Kuipers 86] Kuipers,B.J.: "Qualitative simulation" en Artificial Intelligence Vol.29, pags. 289-338, Elsevier Science Publishers B.V., North-Holland, 1986.

[Kuipers, Berleant 88] Kuipers,B.J., Berleant,D.: "Using incomplete quantitative knowledge in qualitative reasoning" en Proceedings of the National Conference on Artificial Intelligence, AAAI-88, 1988.

[Kuipers, Chiu 87] Kuipers,B.J., Chiu,C.: "Taming intractable branching in qualitative simulation" en Proceedings of the 10th International Joint Conference on Artificial Intelligence, IJCAI-87, Morgan Kaufmann, Milán, Italia, 1987.

[Laird 84a] Laird J.: "Universal Subgoaling". Ph.D. Thesis, Carnegie-Mellon University. Pittsburgh, PA, 1984.

[Laird 84b] Laird,J.E.: "The SOAR2 user's manual", Computer Science Department, Carnegie Mellon University, USA, 1984.

[Laird et al. 87] Laird,J.E., Rosenbloom,P., Newell,A.: "SOAR: An architecture for general intelligence" en Artificial Intelligence, Vol.33, Elsevier Science Publishers B.V., North-Holland, 1987.

[Laird, Newell 83] Laird J., Newell A.: "A universal weak method". Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1983.

[Leler 88] Leler,W.: "Constraint programming languages. Their specification and generation", Addison Wesley Publishing Company, 1988.

[Lenat et al. 83] Lenat D.B., Davis R., Doyle J., Genesereth M., Goldtein I. y Schrobe H.: "Reasoning about reasoning" en "Building Expert Systems", F.Hayes-Roth, D.A. Waterman y D.B. Lenat (Eds), Addison-Wesley, Reading, MA, 1983.

[Lindsay et al. 81] Lindsay R.K., Buchanan B.G., Feigenbaum E.A., Lederberg J J.: "Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project". McGraw Hill, 1981.

[Mackworth 77] Mackworth, A. K.: "Consistency in Networks of Relations" Artificial Intelligence nº 8 Pg: 99-118. 1977

[Mackworth, Freuder 85] Mackworth, A. K., Freuder, E. C.: "The Complexity of Some Polinomial Network Consistency Algrithms for Constraint Satisfaction Problems" Artificial Intelligence nº 25 Pg: 65-74. 1985.

[Martin, Fateman 71] Martin M.A., Fateman R.J.: "The MACSYMA System" Proc 2nd Symp. Symbolic and Algebraic Manipulation. Los Angeles, 1971.

[Mc Carthy 86] Mc Carthy,J.: "Applications of circumscription to formalizing commonsense knowledge" en Artificial Intelligence Vol.28, pags. 89-116, Elsevier Science Publishers B.V., North-Holland, 1986.

[McDermott 82] McDermott J.: "R1: A Rule-Based Configurer of Computer Systems". Artificial Intelligent 19, pag 39-88, 1982.

[McDermott 89] McDermott J.: "Preliminary Steps Toward a Taxonomy of Problem-Solving Methods". Capítulo 8 en "Automating Knowledge Acquisition for Expert Systems". Elsevier Science Publishers B.V., North-Holland, 1989.

[Meyer 88] Meyer,B.: "Object-oriented software construction", Prentice Hall International, Cambridge, MA, USA, 1988.

[Michalski, et al. 86] Michalski,R.S., Carbonell,J.G., Mitchell,T.M.: "Machine Learning: An artificial intelligence approach". Vol. II, Morgan Kaufmann Publishers, 1986.

[Minsky 75] Minsky,M.: "A framework for representing knowledge" en The Psychology of Computer Vision, Winston,P.H. (ed), Mc Graw Hill, Nueva York, NY, USA, 1975.

[Minsky 81] Minsky,M.: "A Framework for representing knowledge" en In Mind Design, Haugeland,J. (ed), The MIT press, Cambridge, MA, USA, 1981.

[Minton 87] Minton,S.: "Strategies of learning search control rules: An explanation based approach" en Proceedings of the 10th International Joint Conference on Artificial Intelligence, IJCAI-87, Morgan Kaufmann Publishers, Milán, Italia, 1987.

[Minton, et al. 88] Minton,S., Knoblock,C.A., Kuokka,D.R., Gil,Y., Carbonell,J.G.: "Prodigy 1.0. The manual and tutorial", Report de la Universidad Carnegie Mellon, Computer Science Department, USA, 1988.

[Mohr, Henderson 86] Mohr,R., Henderson,T.C.: "Arc and path consistency revisited" en Artificial Intelligence, Vol.28, pags. 225-233, Elsevier Science Publishers B.V., North-Holland, 1986.

[Molina 90] Molina,M.: "Proyecto de entorno de representación del conocimiento basado en reglas y marcos", Proyecto de fin de carrera, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, 1990.

[Newell 82] Newell,A.: "The knowledge level" en Artificial Intelligence, Vol.18, pags. 87-127, Elsevier Science Publishers B.V., North-Holland, 1982.

[Newell, Simon 63] Newell A., Simon H.: "GPS: A Program thet Simulates Human Thought". en "Computers and Thougth" Feigenbaum y Feldman (eds). McGraw Hill, 1963.

[Newell, Simon 72] Newell A., Simon H.: "Human Problem Solving". Prentice Hall, 1972.

[Nii 79] Nii H.P., Aiello N.: "AGE (Attempt to Generaliza): A Konowledge-Based Program for building Knowledge-Based Programs". Procc. IJCAI-79, Tokio, pg 645-655, 1979.

[Nii 86a] Nii H.P.: "Blackboard systems". AI Magazine, 7 (3 y 4), pags 38-53 y 82-107, 1986.

[Nii 86b] Nii H.P.: "Blackboard Systems: the Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures". AI Magazine, Vol. 7, No 2,3, 1986.

[Nilsson 80] Nilsson N.J.: "Principles of Artificial Intelligence". Tioga Publising Company, 1980. Versión castellana, Díaz de Santos, 1987.

[Parnas 72] Parnas D.L.: "On Criteria to Be Used in Decomposing Systems into Modules". CACM, Vol.14, No. 1, pag. 221-227, April 1972.

[Pozo 92] Pozo R.:"Modelos de SimulaCión Cualitativa basados en Tareas: un caso práctico". Proyecto Fin de Carrera, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, 1992.

[Pressman 88] Pressman R.: "Ingeniería del Software: un enfoque práctico". 2nd ed. McGraw-Hill, 1988.

[Quinlan 79] Quinlan J.: "Discovering Rules by Induction from Large Collections of Examples". en Expert Systems in the Microelectronic Age. D. Michie ed. Edinburgh University Press, 1979.

[Reddy et.al 73] Reddy .D.R., Erman L.D., Fennell R.D. y Neely R.B.: "The Hearsay Speech Understanding System: An Example of the Recognition Process". en Proc. 3rd int. Joint Conference on Artificial Intelligence, pags. 185-193. Stanford. CA, 1973.

[Robinson 65] Robinson J.A.: "A Machine Oriented Logic Based on the Resolution Principle". Journal ACM (12, 1), 1965.

[Rosenbloom, Newell 86] Rosenbloom P., Newell A.: "The chunking of goal hierarquies: A generalized modelo of practice". en R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds), Machine Learning: An Artificial Intelligent Approach. Morgan-Kaufmann. Los Altos CA, 1986.

[Salmerón 92] Salmerón A.: "Diseño de sistemas de Diagnóstico Profundo Basado en Modelos Cualitativos: Arquitecturas y Métodos". Tesis Doctoral. Facultad de Informática, Universidad Politécnica, Madrid, 1992.

[Schank, Abelson 77] Schank R.C., Abelson R.: "Scripts, Plans and Underestanding". Hilldale J.J. Lawrence Erlbaum and Associates, 1977.

[Shoham 88] Shoham Y.: "Cronological Ignorance: Experiments in Non Monotonic Temporal Reasoning". Artificial Intelligence 36, pag. 279-331, 1988.

[Shortliffe 76] Shortliffe,E.H.: "Computer-based medical consultation: MYCIN", Elsevier, Nueva York, NJ, USA, 1976.

[Smith, Genesereth 85] Smith D., Genesereth M.: "Ordering conjuctive queries". Artificial Intelligence 26, pag 171-215, 1985.

[Steels 87] Steels,L.: "The deepening of expert systems" en AI communications, Vol.0, No.1., 1987.

[Steels 88] Steels,L.: "Second generation expert systems" en Future Generation Expert Systems, Vol.1, No.4, pags. 213-221, 1988.

[Steels 89] Steels,L.: "The deepening of expert systems" en Perspectives in Artificial Intelligence: Volume I Expert Systems: Application and Technical Foundations, Campbell,J.A. y Cuena,J. (eds), Ellis Horwood Limited, pags. 17-29, Chichester, Gran Bretaña, 1989.

[Steels 90] Steels, L.: "Components of Expertise". AI Magazine, pags 28-48. Summer 1990.

[Stefik, Bobrow 83] Stefik M., Bobrow D.G.:"The LOOPS Manual". Xerox PARC, December, 1983,

[Wasserman 83] Wasserman A.: "Information System Methodology". en Software Design Techniques, P.Freeman and A.Wasserman (eds). 4th ed. IEEE Computer Society Press, pag 43, 1983.

[Wielinga 87] Wielinga,B.: "Model-Driven Knowledge Acquisition: Interpretation Models", Memorandum 87 of the VF-Project Knowledge Acquisition in Formal Domains, 1987.

[Wiemer, Sincovec 84] Wierner R., Sincovec R.: "Software Engineering with Modula-2 and Ada". Wiley, 1984.

[Winston 84] Winston P.H.: "Artificial Intelligence, 2nd edition", Addison Wesley, 1984.

[Winston 87] Winston,P.H.: "The commercial debut of artificial intelligence" en Applications of expert systems, Quinlan,J.R. (ed), Addison Wesley, 1987.

ANEJO A: CONSTRUCCION DE UN SISTEMA CON LA ARQUITECTURA PROPUESTA: EL SISTEMA INTELIGENTE DE RAZONAMIENTO HIDROLOGICO (S.I.R.A.H).

El Ministerio de Obras Publicas y Urbanismo español ha llevado a cabo un programa de desarrollo de sistemas de información para su instalación en cuencas hidrográficas. El objetivo de estos sistemas es la generación de bases de datos con información sobre precipitaciones, caudales y niveles que permitan el control y predicción de situaciones de avenida así como la gestión óptima de recursos.

Para conseguir el objetivo de predicción y control debía construirse un sistema experto que, apoyado en esas bases de datos, produjera predicciones de evolución de caudales y niveles de agua junto con recomendaciones para control y protección civil. [Cuena 83] presenta una primera aproximación a la estructura y métodos de construcción para implantar este tipo de sistemas basados en el conocimiento.

Para dar servicio al programa global del Ministerio se planteó el desarrollo de un entorno específico de software para representación del conocimiento. Los principales requerimientos de este entorno eran:

1) Operatividad en entornos hardware y software de propósito general con sistema operativo UNIX.
2) Englobar no sólo los paradigmas de razonamiento para resolución de problemas de tipo uniforme (reglas y marcos) sino también paradigmas de razonamiento profesional utilizados por los hidrólogos.

La respuesta a esos objetivos y requerimientos es el entorno SIRAH. Este anejo presenta un resumen de sus principales características, mostrando el entorno como instancia del concepto general objeto de la tesis y describiendo los módulos tarea para razonamiento cualitativo sobre los componentes físicos más relevantes. Los ejemplos utilizados para la descripción del entorno corresponden a partes de un modelo real construido para la cuenca de Jucar. El modelo completo, y el entorno mismo, se encuentran en el Departamento de Inteligencia Artificial de la Facultad de Informática de Madrid, así como instalados -y en fase de refino con datos reales, en la Confederación Hidrográfica del Jucar (Valencia).

A-I. Definición del problema.

La superficie de una cuenca hidrográfica está compuesta de áreas receptoras de lluvia que, siendo partes de la misma, tienen un único punto de drenaje(1).

(1) Un punto de drenaje es aquel por el que se produce la respuesta en caudal.

Durante una tormenta de lluvia, el caudal de respuesta drenado desde las áreas receptoras entra en la red de canales de drenaje, donde pueden distinguirse:

1) Las redes de transporte en las cuencas altas (barrancos), cuyos cauces tienen fuertes pendientes produciendo, en consecuencia, régimen rápido de transporte de las aguas.
2) El tramo final de río, donde las pendientes son más suaves produciendo, en consecuencia, régimen lento de transporte y efectos de remanso(2) y embalse con riesgo de desbordamiento.

Puede concebirse un sistema de flujo para describir el comportamiento de una cuenca en situación de tormenta: un flujo de lluvia actúa sobre las áreas receptoras, que producen como salida caudales que entran en las redes de transporte rápido, las cuales drenan caudales de entrada en la plana baja o tramo inundable.

A-II. Concepto del sistema.

De acuerdo con este concepto se definen tres tareas básicas de simulación (TBSs):

1) Area Receptora, para modelizar el comportamiento de un área sometida a la acción de la lluvia, produciendo como resultado caudales de entrada para la red de transporte.
2) Redes de Transporte Rápido o Barrancos, para modelizar el comportamiento de las redes de transporte en las cuencas altas, donde los caudales generados en las correspondientes áreas receptoras se concentran produciendo caudales superficiales (con poca o nula infiltración). Estos caudales producen impactos en los tramos inundables del río en la cuenca baja.
3) Tramo Inundable de Río, para modelizar la evolución de los niveles de agua en los tramos inundables de la cuenca baja, sometidos a los diferentes caudales producidos por las redes de la cuenca alta.

Acorde con las definiciones generales dadas en los capítulos 5 al 7, cada TBS se define mediante un marco con:

1) Atributos conclusión, conjunto de posibles valores de variables de estado para su uso por subsiguientes módulos TGE.
2) Una base de conocimiento definida por medio de reglas y restricciones para modelización cualitativa del comportamiento.
3) Un motor de inferencia para razonar, usando la base de conocimiento de comportamiento para deducir posibles valores de los atributos conclusión. Existirá también una base y su correspondiente interprete

(2) El efecto de remanso produce que el nivel de agua en un punto del cauce no dependa solo de los niveles aguas arriba sino también de los niveles aguas abajo.

que opere en fase de creación de la TBS, para deducir valores de características más elaboradas a usar en el proceso de simulación cualitativa, partiendo de atributos más simples de adquirir al definir físicamente la TBS.

Análogamente, y de acuerdo con la estructura general descrita en el capítulo 6, se definen tres tipos de TGE:

1) La tarea de generación de escenarios meteorológicos. Esta tarea infiere combinaciones, posibles y relevantes para los objetivos de control, de predicciones de lluvia local, para actuar sobre el conjunto de áreas receptoras.
2) La tarea de generación de escenarios de entrada a barrancos, que infiere las combinaciones de caudal de entrada a las redes de la cuenca alta.
3) La tarea de generación de escenarios del tramo inundable, que infiere combinaciones de caudales de salida de los barrancos y que inciden en el tramo inundable.

Finalmente, se define un módulo de control para ejecutar la estrategia general y marcar los objetivos de simulación.

A-III. El proceso de razonamiento.

La estrategia de razonamiento basado en estos conceptos constituye una adaptación del esquema general mostrado en el capítulo 5 (figura A.1), distinguiendose las siguientes etapas:

1) Interpretación en linea de los datos recibidos del sistema de información para identificar problemas actuales.
2) Exploración de problemas potenciales futuros, para lo cual una Tarea General de Control (TGC) define patrones de comportamiento final a ser explorados mediante razonamiento predictivo. Por ejemplo, pueden definirse problemas potenciales para su exploración si en algunos puntos los niveles actuales son ya altos aunque todavía no problemáticos, o bien siendo bajos crecen muy rápidamente.
3) Generación, mediante la TGE Meteorológica, de escenarios de lluvia futura sobre áreas receptoras. Estos escenarios se producen en orden de proximidad a la tendencia registrada reciente y únicamente se retiene el primero de ellos para continuar el proceso. Si no se genera ninguno se cede control a la TGC.
4) Búsqueda preferente en profundidad, a base de campañas de simulación, de los diferentes elementos físicos:
    - Se introduce, como entrada del conjunto de TBSs de áreas receptoras, el primer escenario meteorológico producido.

- Se genera un primer escenario de caudales de entrada a barrancos a partir de las hipótesis de caudal de salida obtenidas en áreas receptoras (TGEs de barrancos).

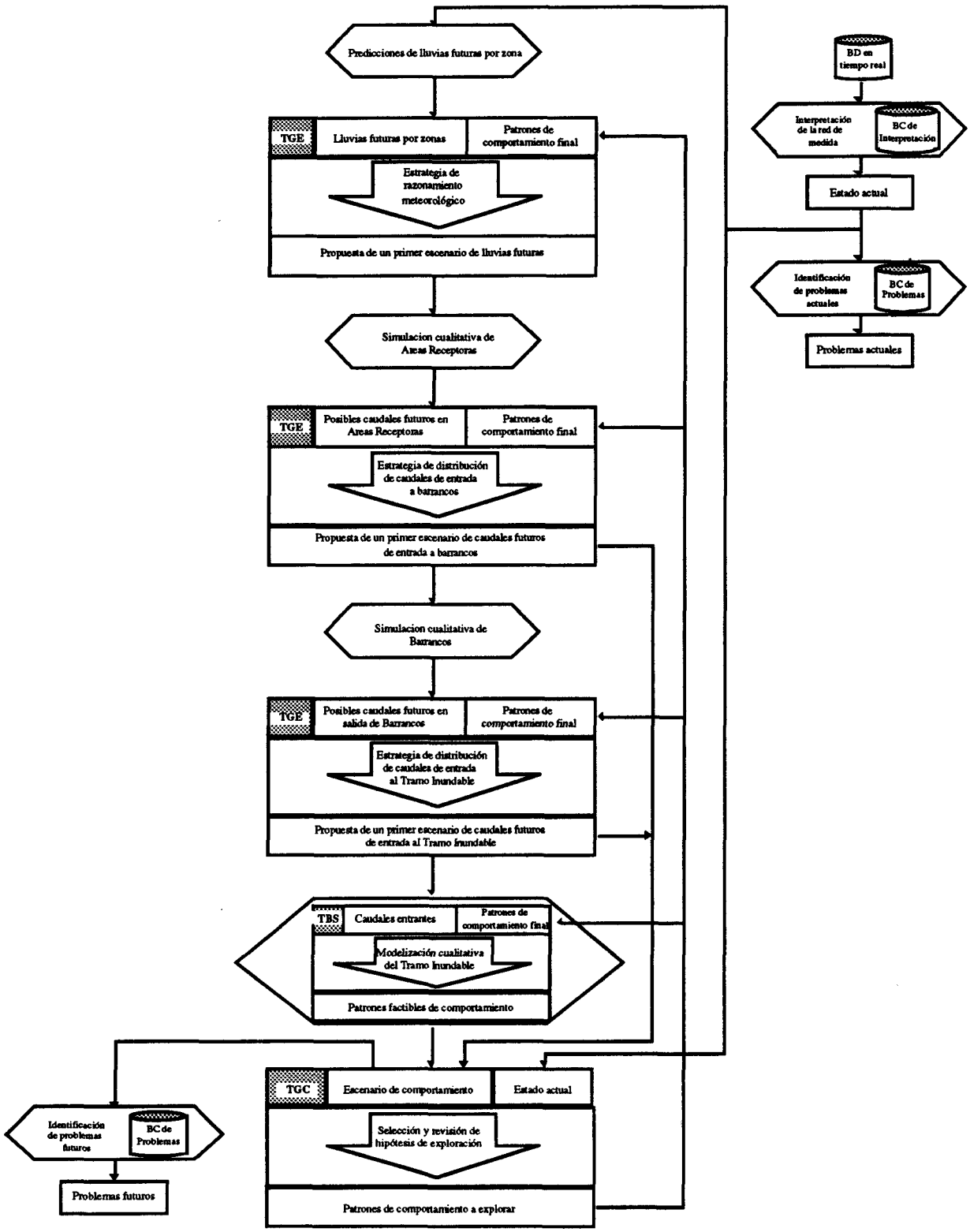


Figura A.1: Estrategia general de razonamiento.

* El escenario de caudales entrantes a barrancos es introducido como entrada a las redes de tramos de transporte de los barrancos.
* Se produce un conjunto de escenarios de caudales de salida en barrancos con objeto de:
  * Identificar problemas en los casos donde sea posible hacerlo directamente a partir de las previsiones de caudal.
  * Actuar como entradas de la Tarea de Simulación del Tramo Inundable, en los casos donde haya información sobre niveles de agua.

Si no se genera ningún escenario se cede control a la TGC.

5) Una vez obtenida una gama de comportamientos (caudales y/o niveles) mediante este proceso se plantea la elección de la siguiente hipótesis de patrones de comportamiento final a explorar. Esto se deduce en la Tarea General de Control mediante un paso de razonamiento apoyado en una base de conocimiento que utiliza como premisas los comportamientos obtenidos y los patrones explorados y pendientes de exploración. La tarea genera restricciones adicionales para las TGEs de manera que, cuando estas TGEs se procesan con las nuevas restricciones, producen un primer escenario que, una vez simulado, dará lugar a comportamientos suficientemente diferentes de los ya inferidos. Con los nuevos escenarios así generados, se realiza un nuevo proceso de búsqueda en profundidad desde el paso 3).

El proceso, en su conjunto, se repite hasta que la Tarea General de Control no emita más hipótesis de patrones a explorar o se supere un tiempo prefijado.

A-IV. Construcción de modelos hidrológicos para predicción de inundaciones.

Construir un modelo del conocimiento para simulación cualitativa del comportamiento de una cuenca hidrográfica, con el entorno SIRAH, requiere los siguientes pasos:

1) Formulación de las bases de conocimiento para inferencia de problemas.
2) Formulación de las bases de conocimiento para planificación de acciones de control de embalses y recomendaciones de protección civil.
3) Definición de un conjunto de instancias de TBSs representando los diferentes tipos de componentes físicos: áreas receptoras, barrancos, tramos de transporte, tramos inundables, etc.
4) Formulación de los elementos TGE para filtrado de comportamientos intermedios.
5) Formulación de la base de conocimiento para la Tarea General de Control, para focalización de la búsqueda y revisión de las alternativas de comportamiento global producidas por simulación.

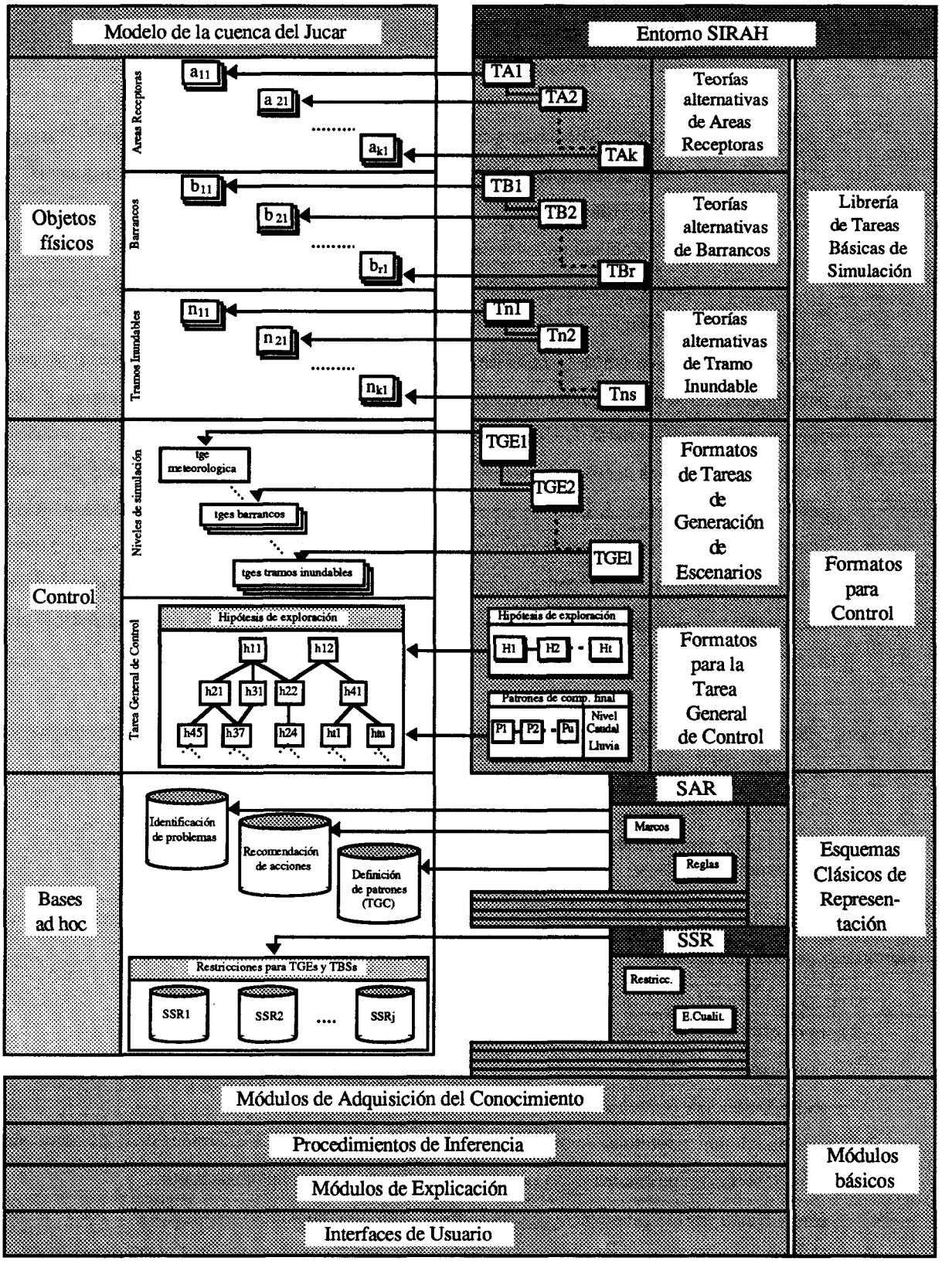


Figura A.2: Construcción de modelos hidrológicos con el entorno SIRAH.

El entorno SIRAH constituye, pues, una instancia del tipo de entorno general propuesto en el capítulo 7. Por consiguiente, proporciona clases de tareas predefinidas (para cada tipo elemento físico de una cuenca) así como procedimientos de soporte para la construcción del modelo. En la figura A.2 se muestra el esquema general del entorno hidrológico.

A-V. Tarea de Generación de Escenarios Meteorológicos.

Las TGEs se definirán como resultado de especificar el concepto general presentado en el capítulo 6. Se muestra a continuación, a modo de ejemplo representativo, la TGE Meteorológica. La estructura general de la tarea se muestra, de forma resumida, en la figura A.3.

Acorde con la estructura general de este tipo de tarea, la TGE meteorológica alberga los atributos siguientes:

1) Predicciones de lluvia por Zona Meteorológica(1), especificándose por cada zona el subdominio o gama de valores de lluvia futuros. Estas predicciones, realizadas anteriormente a la llamada de esta TGE, se producen a nivel de cada zona definida y constituyen los atributos básicos de la TGE.
2) Patrones de nivel y caudal en diferentes puntos, que servirán para identificar escenarios relevantes.
3) Patrón de referencia, compuesto por el conjunto de predicciones de lluvia que se estima más probable. Si la evolución reciente de las lluvias en zonas es conocida, el patrón de referencia será dicha evolución. En caso contrario se elegirá según criterios estratégicos (la evolución más pesimista, evoluciones medias, etc).

Las bases de conocimiento de la TGE meteorológica , para regresión de patrones y consistencia, se definen de la forma siguiente:

1) Base de regresión de patrones, compuesta a su vez de dos subbases:
   - Base de identificación de restricciones activas. Este conocimiento se define por reglas que deducen, a partir de los patrones activos planteados, los umbrales de volúmenes de lluvia a ser alcanzados por diversas agregaciones de zonas para que las predicciones de lluvia en esas zonas se consideren pertinentes. Esta base define, al mismo tiempo, qué restricciones han de tenerse en cuenta, atendiendo a los patrones activos para la regresión.

(1) Una Zona Meteorológica es una superficie de la cuenca en la que se asume comportamiento meteorológico uniforme. Se modelizan también como TBSs, si bien basadas únicamente en formas superficiales de razonamiento. Por esta razón no se incluyen en este anejo.

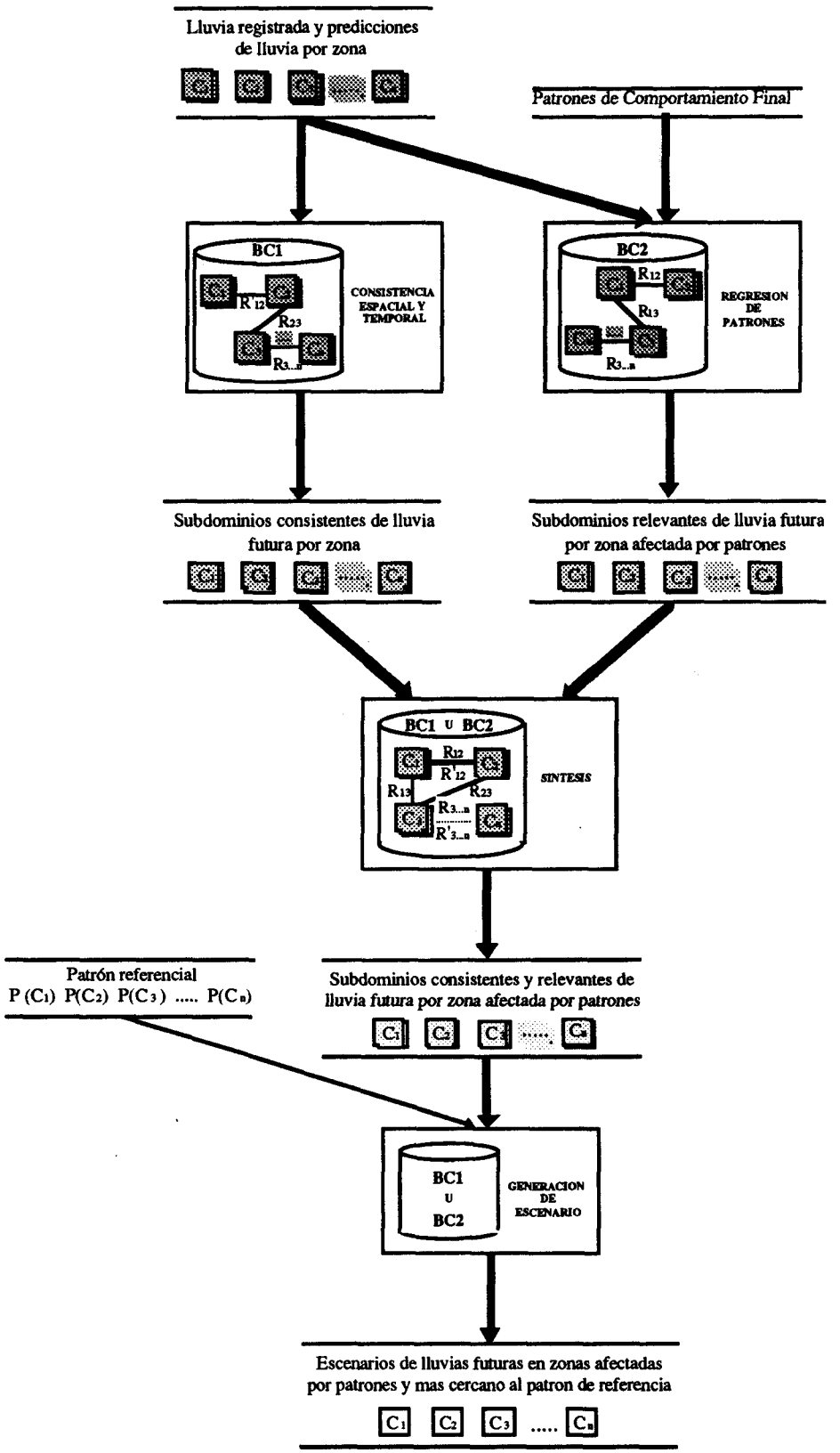


Figura A.3: Estructura de la TGE Meteorológica.

- Base de inferencia de subdominios pertinentes de predicción de lluvias. Este conocimiento se define por restricciones que relacionan predicciones de lluvias en grupos de zonas imponiendo que la suma de dichas predicciones supere (o quede por debajo, en su caso) de umbrales fijados en la base anterior.

La base de regresión, en conjunto, se utiliza para inferir predicciones de lluvia necesarias en zonas para que se cumpla la hipótesis de patrones propuesta.

2) Base de consistencia espacio-temporal, compuesta a su vez por tres subbases:
   - Base de deducción del volumen de lluvia registrado; este conocimiento se define mediante restricciones que relacionan lluvias registradas en zonas y permiten deducir el volumen registrado por grupos de zonas.
   - Base de inferencia de volúmenes máximos y mínimos; este conocimiento se define mediante reglas que deducen, a partir del volumen registrado de lluvia por grupos de zonas y la tendencia futura en esas zonas, umbrales máximos y mínimos de verosimilitud del volumen de lluvia esperado en dichas agregaciones.
   - Base de consistencia; este conocimiento se define mediante restricciones que limitan el crecimiento del volumen de lluvia previsto en zonas según los umbrales máximos y mínimos deducidos por la base anterior.

La estrategia general de la tarea (ver figura A.3) cubre el objetivo de proponer un escenario de lluvia, operando en dos pasos:

1) Restricción de dominios de lluvias futuras en zonas meteorológicas.
2) Generación del primer escenario que cumple las condiciones impuestas de verosimilitud y pertinencia según los objetivos propuestos por la TGC.

La deducción de lluvias posibles en zonas se realiza, a su vez, mediante las siguientes etapas de razonamiento:

1) Regresión de patrones activos (linea de pertinencia). Por cada zona se obtienen sus alternativas de lluvia necesarias (pertinentes) según los objetivos de simulación (patrones) propuestos. Si se obtiene el dominio vacío para alguna zona significará que la hipótesis de patrones de exploración planteada no es factible.
2) Restricción de dominios de lluvias por evaluación de su consistencia espacial y temporal. Por cada zona se obtienen las alternativas de lluvia futura consideradas verosímiles. Nuevamente, si se llega al dominio vacío para alguna zona significará que la hipótesis de patrones propuesta, siendo pertinente, no es verosímil, por lo que debe rechazarse.

3) Síntesis. A partir de las bases de restricciones anteriores se realiza un proceso de satisfacción conjunta que permite alcanzar dominios de lluvias futuras por zona pertinentes y verosímiles de forma simultanea. Si en este paso se obtuviera algún dominio vacío, se rechazaría la hipótesis propuesta por la TGC, pero sin poder especificar por qué criterio, pertinencia por objetivos o verosimilitud, indicándose infactibilidad por ambos criterios aplicados de forma conjunta.

Finalmente, se expone la descripción formal de la TGE Meteorológica en el lenguaje DECON. Nótese como el lenguaje permite la utilización conjunta y transparente de formas de representación basados en reglas y restricciones, todo ello de acuerdo con la estrategia general propuesta.

TGE: Meteorologia **ES UNA** Tge

VARIABLES:

LR lluvia registrada en mm

Alfambra Ademuz

LR Alfambra Ademuz (INTERVALO) [Pondera((Teruel Alfambra) lluvia medida en mm en las ultimas 4 horas,0.33, (Arquillo) lluvia medida en mm en las ultimas 4 horas,0.33, (Turia Ademuz) lluvia medida en mm en las ultimas 4 horas,0.33)]; Alfambra Ademuz Min (INSTANCIA DE Lluvias previstas) [Reglas];

Alfambra Ademuz Max (INSTANCIA DE Lluvias previstas) [Reglas];

Alarcon

LR Alarcon (INTERVALO) [Pondera((La Mancha Alarcon_Alcala) lluvia medida en mm en las ultimas 4 horas,0.33, (Zona de lluvias Contreras) lluvia medida en mm en las ultimas 4 horas,0.33, (Zona de lluvias Alarcon) lluvia medida en mm en las ultimas 4 horas,0.33)];

Alarcon Min (INSTANCIA DE Lluvias previstas) [Reglas];

Alarcon Max (INSTANCIA DE Lluvias previstas) [Reglas];

Cofrentes

LR Cofrentes (INTERVALO) [Pondera((Forata) lluvia medida en mm en las ultimas 4 horas,0.33, (Loriguilla y Buseo) lluvia medida en mm en las ultimas 4 horas,0.33, (Cabriel Contreras_Cofrentes) lluvia medida en mm en las ultimas 4 horas,0.33)];

Cofrentes Min (INSTANCIA DE Lluvias previstas) [Reglas];

Cofrentes Max (INSTANCIA DE Lluvias previstas) [Reglas];

Ribera Alta

LR Ribera Alta (INTERVALO) [Pondera((Cortes) lluvia medida en mm en las ultimas 4 horas,0.33, (Tous) lluvia medida en mm en las ultimas 4 horas,0.33, (Tous_Sellent) lluvia medida en mm en las ultimas 4 horas,0.33)];

Ribera Alta Min (INSTANCIA DE Lluvias previstas) [Reglas];

Ribera Alta Max (INSTANCIA DE Lluvias previstas) [Reglas];

Beniarres

LR Beniarres (INTERVALO) [Pondera((Albaida_Canyoles) lluvia medida en mm en las ultimas 4 horas,0.33, (Beniarres) lluvia medida en mm en las ultimas 4 horas,0.33, (Alto Vinalopo) lluvia medida en mm en las ultimas 4 horas,0.33)];

Beniarres Min (INSTANCIA DE Lluvias previstas) [Reglas];

Beniarres Max (INSTANCIA DE Lluvias previstas) [Reglas];

La Marina

LR La Marina (INTERVALO) [Pondera((Zona de lluvias Marina Baja) lluvia medida en mm en las ultimas 4 horas,0.5, (Amadorio) lluvia medida en mm en las ultimas 4 horas,0.5)];

La Marina Min (INSTANCIA DE Lluvias previstas) [Reglas];

La Marina Max (INSTANCIA DE Lluvias previstas) [Reglas];

La Nao

LR La Nao (INTERVALO) [Pondera((Zona de lluvias Ribera) lluvia medida en mm en las ultimas 4 horas,0.33, (La Safor) lluvia medida en mm en las ultimas 4 horas,0.33, (La Nao) lluvia medida en mm en las ultimas 4 horas,0.33)];

La Nao Min (INSTANCIA DE Lluvias previstas) [Reglas];

La Nao Max (INSTANCIA DE Lluvias previstas) [Reglas];

Ribera Baja

LR Ribera Baja (INTERVALO) [Pondera((Zona de lluvias Ribera) lluvia medida en mm en las ultimas 4 horas,0.33, (Forata_Algemesi) lluvia medida en mm en las ultimas 4 horas,0.33, (Albaida_Canyoles) lluvia medida en mm en las ultimas 4 horas,0.33)];

Ribera Baja Min (INSTANCIA DE Lluvias previstas) [Reglas];

Ribera Baja Max (INSTANCIA DE Lluvias previstas) [Reglas];

Castellon

LR Castellon (INTERVALO) [Pondera((Cabecera ramblas Poyo y Castellana) lluvia medida en mm en las ultimas 4 horas,0.33, (Vega Valencia) lluvia medida en mm en las ultimas 4 horas,0.33, (Plana Castellon) lluvia medida en mm en las ultimas 4 horas,0.33)];

Castellon Min (INSTANCIA DE Lluvias previstas) [Reglas];

Castellon Max (INSTANCIA DE Lluvias previstas) [Reglas];

Poyo

LR Poyo (INTERVALO) [Pondera((Cabecera ramblas Poyo y Castellana) lluvia medida en mm en las ultimas 4 horas,0.33, (Plana Castellon) lluvia medida en mm en las ultimas 4 horas,0.33, (Regajo) lluvia medida en mm en las ultimas 4 horas,0.33)];

Poyo Min (INSTANCIA DE Lluvias previstas) [Reglas];

Poyo Max (INSTANCIA DE Lluvias previstas) [Reglas];

Arenos

LR Arenos (INTERVALO) [Pondera((Regajo) lluvia medida en mm en las ultimas 4 horas,0.33, (Sichar) lluvia medida en mm en las ultimas 4 horas,0.33, (Zona de lluvias Arenos) lluvia medida en mm en las ultimas 4 horas,0.33)];

Arenos Min (INSTANCIA DE Lluvias previstas) [Reglas];

Arenos Max (INSTANCIA DE Lluvias previstas) [Reglas];

Cenia

LR Cenia (INTERVALO) [Pondera((Maria Cristina y Alcora) lluvia medida en mm en las ultimas 4 horas,0.33, (Bajo Cenia_San Miguel) lluvia medida en mm en las ultimas 4 horas,0.33, (Plana Castellon) lluvia medida en mm en las ultimas 4 horas,0.33)];

Cenia Min (INSTANCIA DE Lluvias previstas) [Reglas];

Cenia Max (INSTANCIA DE Lluvias previstas) [Reglas]

Escenario de entrada: lluvias previstas por zonas

ESCENARIO : (Zona de lluvias) Lluvia prevista

Linea de razonamiento de pertinencia

PERTINENCIA :

BASE DE REGLAS: Volumen diferencial patrones

OBJETIVOS:

VD: Volumen diferencial (por patron)

(Patron Manuel)VD; (Patron Xativa)VD; (Patron Albaida)VD;
(Patron Canyoles)VD; (Patron Carlet)VD; (Patron Montroy)VD;
(Patron Bunyol)VD; (Patron Sellent)VD; (Patron Silla)VD;
(Patron Albufera SW)VD; (Patron Escalona)VD;(Patron Ayora)VD;
(Patron Cofrentes)VD; (Patron Villalba)VD; (Patron Cuenca)VD;
(Patron N_420 PK_18.22)VD; (Patron Belmontejo)VD;(Patron Valverde)VD;
(Patron Valdemoro)VD;(Patron Landete)VD; (Patron Mira)VD;
(Patron Aguilar)VD; (Patron Alfambra)VD; (Patron Teruel)VD;
(Patron Albarracin)VD; (Patron Ademuz)VD; (Patron Calles)VD;
(Patron Lliria)VD; (Patron Sot)VD; (Patron Pedralba)VD;
(Patron Ribarroja)VD; (Patron Manises)VD; (Patron Poyo)VD;
(Patron Carraixet)VD; (Patron Puzol)VD; (Patron Altea)VD;
(Patron Oliva)VD;(Patron Valldigna)VD; (Patron Gandia)VD;
(Patron Alcoy)VD; (Patron Villajoyosa)VD; (Patron Campello)VD;
(Patron Alicante)VD; (Patron Elda)VD; (Patron Aspe)VD;
(Patron Elche)VD; (Patron Valbona)VD; (Patron Sarrion)VD;
(Patron Mijares Medio)VD; (Patron CastellonN)VD; (Patron CastellonS)VD;
(Patron La Barona)VD; (Patron Alcora)VD; (Patron Utiel)VD;
(Patron Requena)VD; (Patron Mijares)VD; (Patron Hortunas)VD;
(Patron Motilla)VD; (Patron Quintanar)VD; (Patron Ledanya)VD;
(Patron La Recueja)VD; (Patron Carcelen)VD; (Patron Ballestar)VD;
(Patron Cenia)VD; (Patron Vinaroz)VD; (Patron Pulpis)VD;

(Patron Onda)VD; (Patron Nules)VD; (Patron Bejis)VD;

(Patron Caudiel)VD; (Patron Segorbe)VD; (Patron Sagunto)VD

FIN BASE

BASE DE RESTRICCIONES: Pertinencia de Lluvias Previsibles

ESPACIO CUALITATIVO: Lluvias previstas

RESTRICCIONES:

(Zona de lluvias Ribera) Lluvia prevista >= (Patron Silla) VD;

(Zona de lluvias Ribera) Lluvia prevista >= (Patron Albufera SW) VD;

(Forata_Algemesi) Lluvia prevista >= (Patron Carlet) VD;

(Forata_Algemesi) Lluvia prevista >= (Patron Montroy) VD;

(Forata_Algemesi) Lluvia prevista >= (Patron Bunyol) VD;

(Albaida_Canyoles) Lluvia prevista >= (Patron Manuel) VD;

(Albaida_Canyoles) Lluvia prevista >= (Patron Xativa) VD;

(Albaida_Canyoles) Lluvia prevista >= (Patron Albaida) VD;

(Albaida_Canyoles) Lluvia prevista >= (Patron Canyoles) VD;

(Tous_Sellent) Lluvia prevista >= (Patron Sellent) VD;

(Tous) Lluvia prevista >= (Patron Ayora) VD;

(Tous) Lluvia prevista >= (Patron Cofrentes) VD;

(Tous) Lluvia prevista >= (Patron Escalona) VD;

(Forata) Lluvia prevista >= (Patron Utiel) VD;

(Forata) Lluvia prevista >= (Patron Requena) VD;

(Forata) Lluvia prevista >= (Patron Mijares) VD;

(Forata) Lluvia prevista >= (Patron Hortunas) VD;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista >= (Patron Poyo) VD;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista >= (Patron Lliria) VD;

(Cabecera Carraixet) Lluvia prevista >= (Patron Carraixet) VD;

(Vega Valencia) Lluvia prevista >= (Patron Puzol) VD;

(Loriguilla y Buseo) Lluvia prevista >= (Patron Calles) VD;

(Plana Castellon) Lluvia prevista >= (Patron CastellonN) VD;

(Plana Castellon) Lluvia prevista >= (Patron Onda) VD;

(Plana Castellon) Lluvia prevista >= (Patron Nules) VD;

(Bajo Cenia_San Miguel) Lluvia prevista >= (Patron Vinaroz) VD;

(Bajo Cenia_San Miguel) Lluvia prevista >= (Patron Pulpis) VD;

(Maria Cristina y Alcora) Lluvia prevista >= (Patron La Barona) VD;

(Maria Cristina y Alcora) Lluvia prevista >= (Patron Alcora) VD;

(Zona de lluvias Embalse de Ulldecona) Lluvia prevista >= (Patron Ballestar) VD;

(Regajo) Lluvia prevista >= (Patron Bejis) VD;

(Regajo) Lluvia prevista >= (Patron Caudiel) VD;

(Zona de lluvias Arenos) Lluvia prevista >= (Patron Valbona) VD;

(Zona de lluvias Arenos) Lluvia prevista >= (Patron Sarrion) VD;

(La Safor) Lluvia prevista >= (Patron Valldigna) VD;

(La Nao) Lluvia prevista >= (Patron Altea) VD;

(La Nao) Lluvia prevista >= (Patron Oliva) VD;

(Beniarres) Lluvia prevista >= (Patron Alcoy) VD;

(Bajo Vinalopo) Lluvia prevista >= (Patron Alicante) VD;

(Alto Vinalopo) Lluvia prevista >= (Patron Elda) VD;

(La Mancha Alarcon_Alcala) Lluvia prevista >= (Patron Motilla) VD;

(La Mancha Alarcon_Alcala) Lluvia prevista >= (Patron Quintanar) VD;

(La Mancha Alarcon_Alcala) Lluvia prevista >= (Patron Ledanya) VD;

(La Mancha Alarcon_Alcala) Lluvia prevista >= (Patron La Recueja) VD;

(La Mancha Alarcon_Alcala) Lluvia prevista >= (Patron Carcelen) VD;

(Zona de lluvias Contreras) Lluvia prevista >= (Patron Valdemoro) VD;

(Zona de lluvias Contreras) Lluvia prevista >= (Patron Landete) VD;

(Zona de lluvias Contreras) Lluvia prevista >= (Patron Mira) VD;

(Zona de lluvias Alarcon) Lluvia prevista >= (Patron Villalba) VD;

(Zona de lluvias Alarcon) Lluvia prevista >= (Patron Cuenca) VD;

(Zona de lluvias Alarcon) Lluvia prevista >= (Patron N_420 PK_18.22) VD;

(Zona de lluvias Alarcon) Lluvia prevista >= (Patron Belmontejo) VD;

(Zona de lluvias Alarcon) Lluvia prevista >= (Patron Valverde) VD;

(Teruel Alfambra) Lluvia prevista >= (Patron Aguilar) VD;

(Teruel Alfambra) Lluvia prevista >= (Patron Alfambra) VD;

(Teruel Alfambra) Lluvia prevista >= (Patron Teruel) VD;

(Arquillo) Lluvia prevista >= (Patron Albarracin) VD;

(Turia Ademuz) Lluvia prevista >= (Patron Ademuz) VD;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+ (Loriguilla y Buseo) Lluvia prevista >= (Patron Sot) VD;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+ (Loriguilla y Buseo) Lluvia prevista >= (Patron Pedralba) VD;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+ (Loriguilla y Buseo) Lluvia prevista >= (Patron Ribarroja) VD;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+ (Loriguilla y Buseo) Lluvia prevista >= (Patron Manises) VD;

(Plana Castellon) Lluvia prevista :+ (Regajo) Lluvia prevista >= (Patron Segorbe) VD;

(Plana Castellon) Lluvia prevista :+ (Regajo) Lluvia prevista >= (Patron Sagunto) VD;

(Bajo Cenia_San Miguel) Lluvia prevista :+ (Zona de lluvias Embalse de Ulldecona) Lluvia prevista >= (Patron Cenia) VD;

(Sichar) Lluvia prevista :+ (Zona de lluvias Arenos) Lluvia prevista >= (Patron Mijares Medio) VD;

(Plana Castellon) Lluvia prevista :+ (Maria Cristina y Alcora) Lluvia prevista :+ (Sichar) Lluvia prevista :+ (Zona de lluvias Arenos) Lluvia prevista >= (Patron CastellonS) VD;

(La Safor) Lluvia prevista :+ (Beniarres) Lluvia prevista >= (Patron Gandia) VD;

(Bajo Vinalopo) Lluvia prevista :+ (Alto Vinalopo) Lluvia prevista >= (Patron Aspe) VD;

(Bajo Vinalopo) Lluvia prevista :+ (Alto Vinalopo) Lluvia prevista >= (Patron Elche) VD;

(Zona de lluvias Marina Baja) Lluvia prevista :+ (Alto Vinalopo) Lluvia prevista >= (Patron Campello) VD;

(Zona de lluvias Marina Baja) Lluvia prevista :+ (Amadorio) Lluvia prevista >= (Patron Villajoyosa) VD

FIN RESTRICCIONES

Linea de razonamiento de verosimiltud

VEROSIMILITUD:

BASE DE REGLAS: Inferencia Lluvias Verosimiles

REGLAS:

&./modelos/tgemm/reglas/verosim

OBJETIVOS:

Alfambra Ademuz Min; Alfambra Ademuz Max;

Alarcon Min; Alarcon Max;

Cofrentes Min; Cofrentes Max;

Ribera Alta Min; Ribera Alta Max;

Beniarres Min; Beniarres Max;

La Marina Min; La Marina Max;

La Nao Min; La Nao Max;

Ribera Baja Min; Ribera Baja Max;

Castellon Min; Castellon Max;

Poyo Min; Poyo Max;

Arenos Min; Arenos Max;

Cenia Min; Cenia Max

FIN BASE

BASE DE RESTRICCIONES: Limitacion de Combinaciones

ESPACIO CUALITATIVO: Lluvias previstas

RESTRICCIONES:

(Teruel Alfambra) Lluvia prevista :+ (Arquillo) Lluvia prevista :+
(Turia Ademuz) Lluvia prevista >= Alfambra Ademuz Min;
(Teruel Alfambra) Lluvia prevista :+ (Arquillo) Lluvia prevista :+
(Turia Ademuz) Lluvia prevista <= Alfambra Ademuz Max;
(La Mancha Alarcon_Alcala) Lluvia prevista :+
(Zona de lluvias Contreras) Lluvia prevista :+ (Zona de lluvias Alarcon) Lluvia prevista <= Alarcon Min;
(La Mancha Alarcon_Alcala) Lluvia prevista :+ (Zona de lluvias Contreras) Lluvia prevista :+ (Zona de lluvias Alarcon) Lluvia prevista <= Alarcon Max;
(Forata) Lluvia prevista :+ (Loriguilla y Buseo) Lluvia prevista :+
(Cabriel Contreras_Cofrentes) Lluvia prevista >= Cofrentes Min;
(Forata) Lluvia prevista :+ (Loriguilla y Buseo) Lluvia prevista :+
(Cabriel Contreras_Cofrentes) Lluvia prevista <= Cofrentes Max;
(Cortes) Lluvia prevista :+ (Tous) Lluvia prevista :+
(Tous_Sellent) Lluvia prevista >= Ribera Alta Min;
(Cortes) Lluvia prevista :+ (Tous) Lluvia prevista :+
(Tous_Sellent) Lluvia prevista <= Ribera Alta Max;
(Albaida_Canyoles) Lluvia prevista :+ (Beniarres) Lluvia prevista :+
(Alto Vinalopo) Lluvia prevista >= Beniarres Min;
(Albaida_Canyoles) Lluvia prevista :+ (Beniarres) Lluvia prevista :+
(Alto Vinalopo) Lluvia prevista <= Beniarres Max;
(Zona de lluvias Marina Baja) Lluvia prevista :+
(Amadorio) Lluvia prevista >= La Marina Min;
(Zona de lluvias Marina Baja) Lluvia prevista :+
(Amadorio) Lluvia prevista <= La Marina Max;
(Zona de lluvias Ribera) Lluvia prevista :+ (La Safor) Lluvia prevista :+
(La Nao) Lluvia prevista >= La Nao Min;
(Zona de lluvias Ribera) Lluvia prevista :+ (La Safor) Lluvia prevista :+
(La Nao) Lluvia prevista <= La Nao Max;
(Zona de lluvias Ribera) Lluvia prevista :+ (Forata_Algemesi) Lluvia prevista :+ (Albaida_Canyoles) Lluvia prevista >= Ribera Baja Min;

(Zona de lluvias Ribera) Lluvia prevista :+ (Forata_Algemesi) Lluvia prevista :+ (Albaida_Canyoles) Lluvia prevista <= Ribera Baja Max;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+

(Vega Valencia) Lluvia prevista :+ (Plana Castellon) Lluvia prevista >= Castellon Min;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+

(Vega Valencia) Lluvia prevista :+ (Plana Castellon) Lluvia prevista <= Castellon Max;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+

(Plana Castellon) Lluvia prevista :+(Regajo) Lluvia prevista >= Poyo Min;

(Cabecera ramblas Poyo y Castellana) Lluvia prevista :+

(Plana Castellon) Lluvia prevista :+(Regajo) Lluvia prevista <= Poyo Max;

(Regajo) Lluvia prevista :+ (Sichar) Lluvia prevista :+

(Zona de lluvias Arenos) Lluvia prevista >= Arenos Min;

(Regajo) Lluvia prevista :+ (Sichar) Lluvia prevista :+

(Zona de lluvias Arenos) Lluvia prevista <= Arenos Max;

(Maria Cristina y Alcora) Lluvia prevista :+

(Bajo Cenia_San Miguel) Lluvia prevista :+ (Plana Castellon) Lluvia prevista >= Cenia Min;

(Maria Cristina y Alcora) Lluvia prevista :+ (Bajo Cenia_San Miguel) Lluvia prevista :+ (Plana Castellon) Lluvia prevista <= Cenia Max

FIN RESTRICCIONES

PATRON REFERENCIA: (Zona de lluvias) Tendencia

FIN TGE

A-VI. Tareas Básicas de Simulación de Areas Receptoras y Barrancos.

Las TBSs para áreas receptoras y barrancos realizan una predicción de respuesta en caudal para un horizonte temporal futuro de varias horas(1), a base de iteraciones de una tarea elemental que infiere, para un periodo básico Δt(2), los posibles nuevos valores de las variables de estado. La estructura general de la tarea se muestra en la figura A.4.

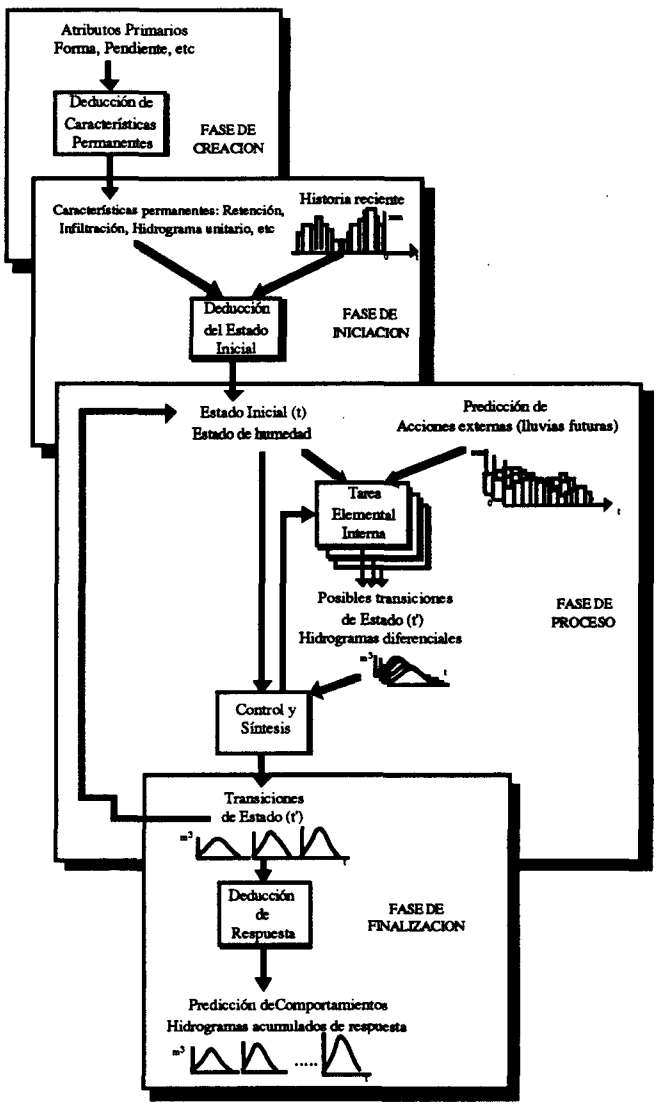


Figura A.4: Estructura de la TBS de Areas Receptoras.

(1) El horizonte real de predicción viene dado por los horizontes individuales de cada tarea. El tiempo medio en la cuenca del Jucar es de 4-5 horas.

(2) Δt son 15 minutos en la cuenca del Jucar, mínimo tiempo de propagación de una onda en esa cuenca. Este parámetro, y algunos otros existentes del mismo tipo, son modificables en el lenguaje DECON.

En las áreas receptoras, el modelo profundo es una versión cualitativa de los modelos tradicionales utilizados por los hidrólogos siendo posible, en este caso, expresar el modelo mediante transiciones del tipo explicito, ya comentadas en el capítulo 5:

$$E(t), A(t, t+\Delta t) ==> E(t+\Delta t')$$

siendo E(t) el estado del área en t y A(t, t+Δt) el conjunto de predicciones de lluvia sobre el área durante el periodo [t, t+Δt]. La linea general de razonamiento (tarea elemental interna) es como sigue:

- La acción externa durante Δt es la lluvia.
- Las variables de estado son la humedad del suelo y el exceso de agua producido, una vez que una parte de la misma se ha infiltrado. Ese exceso producirá escorrentía superficial.
- A partir de los atributos permanentes del área receptora y la lluvia total en Δt se deducen nuevos valores para el estado de humedad y el exceso.
- Se obtiene el exceso neto, a partir del nivel de infiltración y el total de lluvia.
- El exceso neto se supone con distribución uniforme en la superficie del área receptora. Con esta hipótesis, y dada la forma y pendiente del área puede deducirse como característica permanente la distribución en el tiempo de una unidad de exceso de lluvia, utilizando otro parámetro físico del área denominado hidrograma unitario [Chow et al. 88]; es decir, el hidrograma unitario representa la respuesta en caudal del área, a lo largo del tiempo, a un exceso de lluvia unitario en el momento inicial (ver figura A.5).

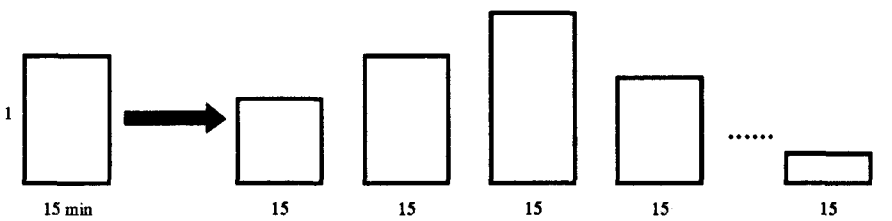


Figura A.5: Hidrograma unitario del exceso.

La distribución a lo largo del tiempo del exceso de lluvia r se obtiene por factorización del hidrograma cualitativo unitario definido por valores mínimo y máximo en cada intervalo Δt.

El procedimiento de inferencia de simulación de un área receptora itera este procedimiento básico utilizando como entrada el hidrograma acumulado hasta el momento (inicialmente el hidrograma plano) y deduciendo, en cada periodo Δt, el hidrograma adicional generado por la tarea de inferencia elemental.

Dado que tanto la entrada como los parámetros físicos se definen de forma aproximada, por intervalos de valores, puede generarse un árbol de posibles evoluciones futuras cuyos niveles corresponden a estados cada Δt. Para

restringir la expansión combinatoria se introduce, en la base de control de la tarea elemental, un paso de síntesis de los comportamientos previstos cada Δt, de tal forma que únicamente se retienen un número limitado de opciones de predicción en cada nivel (la opción mas pesimista, una opción media, la más pesimista, etc), cada una de las cuales definida por un intervalo de valores posibles de respuesta.

En la tarea de simulación del barranco se presenta un caso similar, donde la tarea elemental es la simulación de un tramo de la red arboriforme de cauces que drena el barranco. En este caso la iteración se basa en un recorrido del árbol de tramos, de manera que para cada uno se simula el periodo completo (posibles predicciones de caudal en la cola se infieren de una predicción en cabecera) y los resultados se transmiten a los tramos situados aguas abajo.

En ambos casos, áreas receptoras y barrancos, los parámetros utilizados para simulación de comportamiento se deducen de las características básicas de los elementos físicos mediante un paso de razonamiento en tiempo de creación, apoyado en bases de conocimiento específicas basadas en reglas. Los elementos de conocimiento, para ambos tipos de tareas, pueden definirse como versiones cualitativas de métodos clásicos en hidrología (tareas elementales) o basados en el juicio personal de los profesionales (la base de creación y la de control y síntesis).

Finalmente, a modo ilustrativo, se expone la definición formal en lenguaje DECON de la clase de TBS de áreas receptoras.

TBS Area receptora **SUBCLASE DE** TBS Temporal

ENTRADAS TBS:

Prevision lluvias (INSTANCIAS DE Sucesion binaria)

SALIDAS TBS:

Hidrograma de respuesta (INSTANCIAS DE Sucesion binaria)

[Descomposicion de hidrogramas(()Hidrograma global)]

PROPIEDADES:

dt = 0.5;

alfa1 (NUMERICO) = -0.5; alfa2 (NUMERICO) = 0.0; alfa3 (NUMERICO) = 0.5;

Superficie (NUMERICO) = 1.0;

Tiempo de concentracion (NUMERICO) = 1.0;

Pendiente (NUMERICO) = 3;

Forma de la cuenca (alargada,equilibrada,abanico) = equilibrada;

Capacidad de almacenamiento (muy baja,baja,media,alta) = alta;

Capacidad de infiltracion (muy baja,baja,media,alta) = alta;

Permeabilidad (baja,media,alta) = alta;

Vegetacion (nula,escasa,media,alta)= nula;

Grado de urbanizacion (nulo,bajo,medio,alto)= nulo;

Tipo de vegetacion dominante (bosque,matorral,cultivos,labor) = cultivos;

Densidad de drenaje (nula,baja,media,alta,muy alta) = media

VARIABLES INTERNAS:

Propiedades deducidas

Coeficiente de exceso (INTERVALO) [Reglas];

Valor inicial rama ascendente (lento,normal,rapido) [Reglas];

Valor inicial rama descendente (suave,normal,brusco) [Reglas];

Duracion inicial hidrograma formal (muy pequena,pequena,normal,grande,muy grande) [Reglas];

Valor final rama ascendente (lento,normal,rapido) [Reglas];

Valor final rama descendente (suave,normal,brusco) [Reglas];

Duracion final hidrograma formal (muy pequena,pequena,normal,grande,muy grande) [Reglas];

Tiempo punta hidrograma unitario adimensional (INTERVALO) [Reglas];

Tiempo inflexion hidrograma unitario adimensional (INTERVALO) [Reglas];

Caudal inflexion hidrograma unitario adimensional (INTERVALO) [Reglas];

Tiempo fin hidrograma unitario adimensional (INTERVALO) [Reglas];

Hidrograma unitario (INSTANCIAS DE Hidrograma) [Calculo hidrograma unitario(()Superficie, ()Tiempo de concentracion, ()Tiempo punta hidrograma unitario adimensional, ()Tiempo inflexion hidrograma unitario adimensional, ()Caudal inflexion hidrograma unitario adimensional, ()Tiempo fin hidrograma unitario adimensional, () dt)];

Variables para deducción de estados iniciales

Lluvia problema (INSTANCIA DE Lluvias previstas) [Caracteriza sucesion(()Prevision lluvias,3.6,[0,4])];

Intensidad de lluvia en t (INTERVALO) [Valor instantaneo(() Prevision lluvias, ()Ts)];

Precipitacion en dt (INTERVALO) [Multiplica intervalos(()Intensidad de lluvia en t, () dt)];

Estado de humedad (muy bajo,bajo,medio,alto,muy alto) [Reglas];

Intensidad de infiltracion potencial (INTERVALO) [Reglas];

Precipitacion neta (INTERVALO) [Resta intervalos(()Intensidad de lluvia en t, ()Intensidad de infiltracion potencial)];

Intensidad exceso libre (INTERVALO) [Multiplica intervalos(()Coeficiente de exceso, ()Precipitacion neta)];

Volumen exceso libre (INTERVALO) [Multiplica intervalos(()Intensidad exceso libre, () dt)];

Hidrograma diferencial (INSTANCIAS DE Hidrograma) [Hidrograma por intervalo(()Hidrograma unitario, ()Volumen exceso libre)];

Hidrograma global (INSTANCIAS DE Hidrograma) [Suma y sintesis(()Hidrograma global [ANTES], ()Hidrograma diferencial, ()alfa1, ()alfa2, ()alfa3)];

Precipitacion acumulada (INTERVALO) [Suma intervalos (()Precipitacion acumulada [ANTES], ()Precipitacion en dt)]

VARIABLES ESTADO:

Precipitacion acumulada;

Hidrograma global

CREACION

TBS BASE DE REGLAS Hidrograma Unitario

OBJETIVOS

Coeficiente de exceso;

Hidrograma unitario

FIN BASE

INICIO

TBS BASE DE REGLAS Inicial

OBJETIVOS

Lluvia problema;

Color;

Precipitacion acumulada [Integra sucesion(()Prevision lluvias, ()Tid, ()Tis)];

Hidrograma global [Asigna (()Hidrograma plano)]

FIN BASE

PROCESO

TBS BASE DE REGLAS Humedad

OBJETIVOS

Intensidad de lluvia en t;

Precipitacion acumulada; Estado de humedad ;

Intensidad de infiltracion potencial; Precipitacion neta ;

Intensidad exceso libre;

Hidrograma diferencial;

Hidrograma diferencial [Desplaza(()Ts)];

Hidrograma diferencial [Recorta sucesion (4)];

Hidrograma global

FIN BASE

FINAL

TBS BASE DE REGLAS Respuesta

OBJETIVOS

Hidrograma de respuesta

FIN BASE

FIN TBS

A-VII. Tarea Básica de Simulación de Tramo Inundable.

Para la TBS del tramo inundable se han construido dos versiones: un modelo profundo del comportamiento del río y un modelo superficial (más eficiente) basado en la representación explícita de las transiciones de estado. A continuación se describen ambas alternativas.

Modelo profundo del tramo Inundable.

La TBS del Tramo Inundable se construye, en este caso, mediante un modelo cualitativo creado en base a explorar transiciones de estado implícitas al conjunto de ecuaciones diferenciales constitutivas de un modelo clásico hidrológico: las ecuaciones de Saint Venant del régimen variable [Chow et al. 88] y utilizando ideas de modelización cualitativa de flujos basada en ejes de [Cuena 88].

La TBS del tramo inundable se define mediante un marco con los siguientes elementos:

1) Atributos físicos del cauce: rugosidad, sección transversal en cada perfil(1) a diferentes distancias a lo largo del eje del cauce, etc.
2) Acciones externas, correspondientes a los caudales predecibles pertenecientes al escenario producido por la correspondiente TGE, a partir de los caudales de salida de barrancos.
3) Atributos de estado (nivel y caudal en cada perfil).
4) Patrones de comportamiento final a explorar. Estos son los mismos conceptos utilizados en la TGE pero, en este caso, los patrones no constan únicamente de un atributo sino de una conjunción de valores de variables de estado en un conjunto de perfiles adyacentes.

La base de conocimiento para predicción del comportamiento a partir de un estado inicial se define a partir de las ecuaciones de Saint Venant:

- Ecuación de continuidad:

$$\frac{dQ}{dx} + T\frac{dy}{dt} = q$$

- Ecuación de equilibrio dinámico:

$$2V\frac{\delta Q}{\delta x} + \frac{\delta Q}{\delta t} + (gA - V^2T)\frac{\delta y}{\delta x} = V^2\frac{\delta A}{\delta x}\Big|_y + gA(S_0 - S_t)$$

(1) Un perfil es un elemento de sección transversal en el cauce, donde se miden niveles y caudales.

donde $Q(x,t)$, $V(x,t)$, $A(x,t)$, $y(x,t)$ indican respectivamente volumen, velocidad, área mojada y calado en t en la sección cuya abcisa respecto del eje es x. $T(x,t)$ es el ancho de lámina libre en sección x. S_0 es la pendiente geométrica y:

$$Sf = \frac{n^2 V^2}{R^{4/3}}$$

es la pendiente dinámica. n es el coeficiente de rugosidad y R es el radio hidráulico.

$$\frac{\delta A}{\delta x}\Big|_y$$

es la derivada de la sección a lo largo del eje para calado constante (esta derivada toma valor nulo en cauces perfectamente prismáticos) y $q(x,t)$ es la aportación de caudal externo (cuantitativo) por unidad de longitud.

Las ecuaciones se pueden procesar mediante un esquema en diferencias finitas resultando dos ecuaciones lineales que relacionan los valores de variables en un conjunto de perfiles a lo largo del eje a distancia Δt:

- Ecuación de continuidad:

$$\frac{1}{\Delta x}\left[\Delta Q_{i+1} - Q_i\right] + \frac{T_i + T_{i+1}}{4}\left[\Delta y_{i+1} - y_i\right] = q - \frac{Q_{i+1}+Q_i}{\Delta x}$$

- Ecuación de equilibrio dinámico:

$$\frac{N}{\Delta x}\left[\Delta Q_{i+1} - Q_i\right] + \left[\frac{1}{2\Delta t} - \frac{L\,Sf_{i+1}}{Q_{i+1}}\right]\Delta Q_{i+1} + \left[\frac{1}{2\Delta t} - \frac{L\,Sf_i}{Q_i}\right]\Delta Q_i$$

$$\frac{M}{\Delta x}\left[\Delta y_{i+1} - y_i\right] + \frac{L\,Sf_{i+1}}{A_{i+1}}\,DIF_i\,\Delta y_{i+1} + \frac{L\,Sf_i}{A_i}\,DIF_i\,\Delta y_i = S + L\left[S_0 - \frac{1}{2}(Sf_i + Sf_{i+1})\right]$$

siendo:

y_i, q_i = nivel y caudal en perfil *i*

$L = (gA_i + gA_{i+1})^{1/2}$

$M = 1/2\ ((gA - V^2T)_i + (gA - V^2T)_{i+1})$

$N = (V_i + V_{i+1})/2$

$DIF = -5/3\ T + 2/3\ R\ dP/dy$

Los valores de variables en los coeficientes L, M, N, DIF corresponden al momento *t*.

A efectos de la modelización cualitativa, se definen dos confluencias mediante análisis de signos de los coeficientes en las ecuaciones en diferencias finitas:

- Confluencia de continuidad: $dvQ_j + dym_j = Dq_j \quad (ym_j = (y_j + y_{j+1})/2)$
  donde Dq_j es el flujo neto en el tramo *j* (entre perfiles *j,j+1*) con valores en el espacio cualitativo <-,0,+>.
- Confluencia de equilibrio dinámico: $dvQ_i + dvy_i = dQ_{i+1} + dy_i + Dd_i$
  siendo Dd_i una medida del equilibrio energético en la situación actual con valores en el espacio <-,0,+>.
- Confluencias de definición: $dy_{i+1} - dy_i = dvy_i$
  $dQ_{i+1} - dQ_i = dvQ_i$
  $Dd_{i+1} + dy_i = dym_i$
  (vQ_i y vy_i son las variaciones de caudal y nivel entre perfiles *i,i+1* en el mismo instante).

Estas confluencias, que relacionan los signos de cambio de las variables de estado, se utilizan para inferir el comportamiento futuro del tramo inundable utilizando las siguientes bases de conocimiento:

B1- Base para inferencia del estado inicial (niveles y caudales) en aquellos perfiles donde no se disponga de medidas procedentes del sistema de información. La inferencia puede hacerse utilizando datos disponibles en otros perfiles, así como reglas extraídas de una librería de casos construida aparte, como resultado de realizar simulaciones cuantitativas que se apoyan en el esquema en diferencias finitas.

B2- Base para inferencia de los valores de Dq y Dd para cada tramo entre perfiles sucesivos. Esta base es una versión cualitativa de las fórmulas que evalúan estos valores en el esquema en diferencias finitas y pueden definirse formalmente mediante relaciones entre atributos en la biblioteca de módulos de SIRAH.

B3- Base de confluencias unión de los conjuntos de confluencias de cada tramo (entre perfiles) y donde se introducen los signos deducidos para Dq y Dd.

B4- Base para propagación de las variaciones de flujo a lo largo del eje. Dados los incrementos de flujo Dq en las entradas del tramo y en un horizonte temporal, esta base permite estimar los posibles incrementos de flujo en el mismo horizonte temporal en varios perfiles a lo largo del eje. La base es una abstracción cualitativa de las condiciones de propagación de una onda a lo largo del cauce y puede especificarse mediante reglas genéricas entre atributos predefinidos.

El motor de inferencia utiliza las bases anteriores de la forma siguiente:

1) A partir de los datos existentes como premisa, la base B1 estima, para cada perfil, posibles estados iniciales. Utilizando los patrones de comportamiento final se definen valores incrementales a ser tenidos en cuenta en el análisis futuro con las bases B3 y B4.
2) La base B2 estima los signos de Dd utilizando como premisas los estados iniciales factibles deducidos por la base B1. Los desequilibrios volumétricos Dq se estiman en signo y cantidad a partir de las entradas procedentes de los barrancos y los estados iniciales (Dq puede ser negativo si hay desbordamientos).
3) La base B3 aplica un proceso de satisfacción de restricciones estructurado en dos pasos:
    * Un primer paso de definición de dominios de atributos que sea consistente con el conjunto global de confluencias.
    * Un segundo paso de generación de escenarios de comportamiento combinando los dominios obtenidos y teniendo en cuenta (a) la base de confluencias B3, (b) la proximidad a la tendencia registrada [Shoham 88] y (c) los patrones diferenciales de comportamiento final a explorar. El uso de patrones diferenciales permite la poda de comportamientos que, siendo consistentes, sean irrelevantes para los objetivos. Se define un área de influencia reducida para cada patrón que incluye uno o dos perfiles aguas arriba y dos perfiles aguas abajo. Se examinarán solo aquellas combinaciones que satisfacen el patrón, utilizando aquella parte de la base B3 que afecta al área de influencia definida e imponiendo como condiciones de contorno los dominios de valores obtenidos en el paso anterior.
4) Finalmente, la base B4 deduce los posibles nuevos estados, a partir de la gama de comportamientos obtenidos en (3), los estados iniciales y los incrementos de flujo.

Modelo superficial del Tramo Inundable.

La representación del comportamiento del tramo inundable consiste en definir el modo en el que se realiza el siguiente paso de inferencia:

$$E_0, A_{01} \text{ --> } E_1$$

donde E_0 es el estado actual del tramo inundable considerado como el conjunto de niveles y caudales en secciones transversales significativas y en las bolsas de desbordamiento lateral, A_{01} es el entorno de acciones externas durante la próxima hora definidas como caudales en los afluentes del tramo, y E_1 es el estado del tramo al que se llega al final de la próxima hora.

Dado que el objetivo del estudio del comportamiento del tramo inundable es predecir los problemas de inundación que se pueden presentar, puede realizarse una simplificación importante en el número de estados posibles a considerar, de forma que se planteen los estados que difieran unos de otros en los efectos de inundación que producen. Esta posibilidad permite representar el conocimiento sobre el comportamiento del tramo como un espacio de estados en el que se refleje la forma de transitar entre ellos según las distintas acciones externas. El conocimiento que es preciso representar es el siguiente:

- Caracterización de los estados posibles del tramo inundable.
- Caracterización de las acciones externas posibles a considerar sobre el tramo.
- Definición de la forma de transitar entre los estados a partir de las diversas acciones.

La representación anterior permitirá conocer cómo se comporta el tramo a partir de un cierto estado inicial y unas ciertas acciones externas. Es preciso, por ello, expresar además cómo se selecciona dicho estado inicial y las acciones futuras. El estado inicial se determinará a partir de las medidas que proporcionen los sensores instalados en el río. Sin embargo, este proceso de selección no es inmediato debido a la escasez de medidas existentes. Por ello la determinación del estado inicial proporcionará como respuesta un conjunto de estados candidatos compatibles con el estado medido. Por otra parte, las acciones externas que se producen en el futuro tendrán como procedencia las previsiones de caudales que se efectúan en los barrancos a partir del modelo del río que tiene el entorno SIRAH. Dichas previsiones son, al igual que la selección del estado inicial, de naturaleza no determinista ya que la predicción de caudales en barrancos se compone, en general, de un conjunto de propuestas que es preciso identificar con las acciones externas a considerar en el estudio del comportamiento del tramo.

El esquema elemental de proceso se resume en la figura adjunta A.6. Cada uno de los estados problema considerados del tramo inundable se caracteriza con un conjunto de atributos numéricos representativos de caudal y nivel en puntos significativos (figura A.7). Dicha clase tendrá tantas instancias como estados del tramo inundable se consideren. El conjunto de estos estados se confeccionará para representar el total de situaciones de crecida que el tramo pueda presentar distinguiendo unos de otros por la naturaleza de los problemas que puedan

originar y la cantidad de agua que transporte el cauce del tramo. Nótese que el objetivo será predecir problemas en situación de inundación, por lo que estados que no aporten información en este sentido no serán considerados.

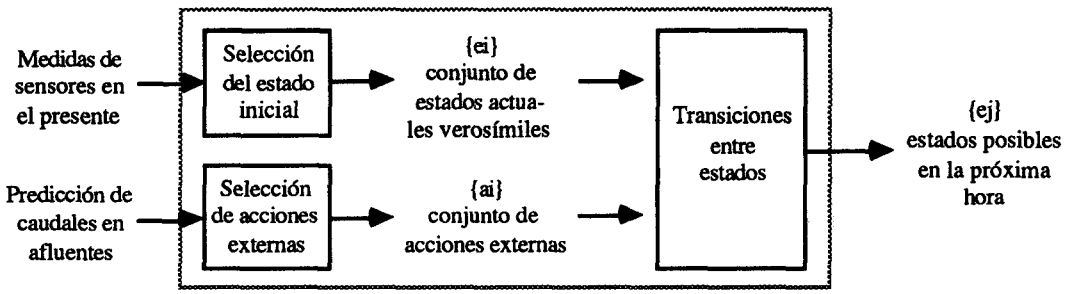


Figura A.6: Esquema general de inferencia del tramo inundable.

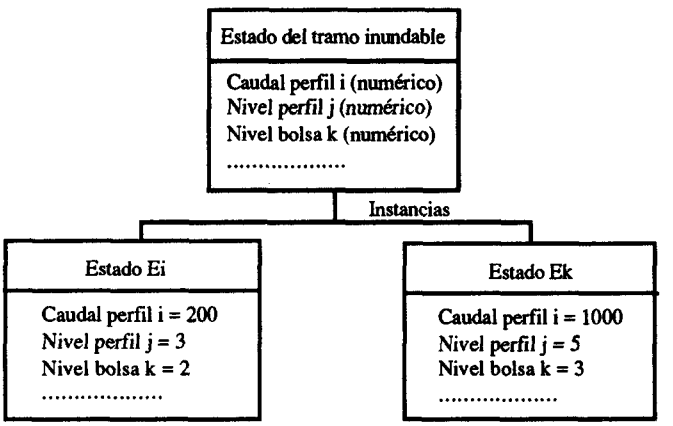


Figura A.7: Definición de Estado del tramo inundable.

La representación de las acciones que inciden sobre los estados se realiza de igual manera que en los estados. Se define una clase que tiene como atributos los caudales en los afluentes del tramo. Sus instancias serán situaciones posibles a considerar de entrada de agua en el tramo (figura A.8).

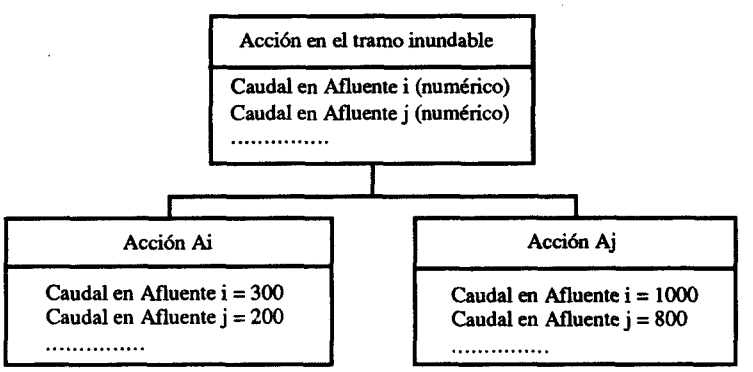


Figura A.8: Acciones externas al tramo inundable.

La selección del estado del tramo inundable consiste en elegir el estado o estados más cercanos a la situación que se está recibiendo por la red de telemedida. Como resultado de la elección se tendrán uno o más estados como actuales en el río. La selección de las acciones que inciden sobre el estado o estado actuales se efectuará de igual manera que en la selección del estado. las acciones externas son resultado del proceso de predicción en los barrancos y se tienen para diversos períodos de tiempo, 1h, 2h, 3h, etc. Habrá, para cada uno de estos períodos, una selección de las acciones posibles dentro del conjunto de acciones establecidas por las instancias de la clase *acción en el tramo inundable.*

Las transiciones de estado se representa de forma explícita mediante reglas. Habrá tantas reglas como transiciones se consideren y se *dispararán* tantas reglas como transiciones se *activen.* El conocimiento así expresado se refiere a transiciones de intervalo de tiempo de 1 hora, es decir, dado un estado en el presente y las acciones a lo largo de la próxima hora, se deduce el estado en la hora siguiente. Este conocimiento debe aplicarse tantas veces como horas se quiera alcanzar en predicción.

A partir de los elementos anteriores el objetivo es obtener las predicciones del estado del tramo en cada una de las próximas horas de predicción. Cabe plantearse la estructura siguiente: Se define un marco de control clase para predicción del estado del tramo inundable que cuenta con el conocimiento global común para cada una de las predicciones (a 1h, 2h, ...) de los estados. Se denominará a esta clase "tramo inundable" y contiene el conocimiento de cómo transitar entre estados, es decir, las reglas que deducen los estados siguientes a partir de los anteriores y las acciones. La clase tramo inundable se define con tres atributos: estado anterior, acción y estado siguiente. A este nivel se definen las reglas de transición (figura A.9).

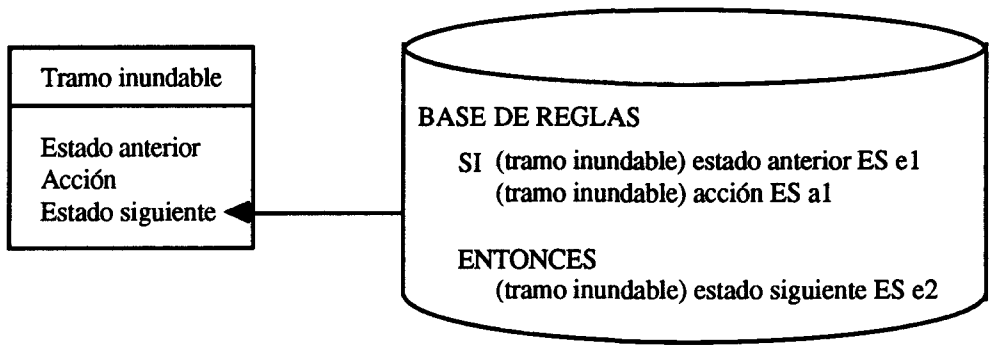


Figura A.9: Clase de Tramo Inundable y reglas de transición entre estados.

Cada predicción (a 1h, 2h, ...) será una instancia o particularización de la clase anterior especificando en la misma la forma de obtener el estado anterior y la acción. De esta forma se tendrá una instancia del tramo inundable a 1h. que tenga en el atributo *estado anterior* el método de obtención del estado inicial mediante un marco de control. Esta instancia definirá también cómo deducir las acciones a 1h. con el mismo marco de control. Para deducir el estado siguiente se heredará el conocimiento en reglas de la clase superior y como resultado de su evaluación

dicho atributo contendrá los posibles estados en la próxima hora del tramo inundable. Para la predicción a 2h. se tendrá como estado anterior el considerado siguiente en la predicción a 1h. En el caso de acción se procede de igual forma que en la predicción a 1h. Para deducir el estado siguiente se heredará, al igual que en la instancia de predicción a 1h., el conocimiento en reglas de la clase superior. De igual forma se opera con las predicciones a 3h, 4h, etc.

Por tanto, en las instancias se especifica cómo deducir el estado anterior y las acciones con ayuda de marcos de control, y en la clase se expresa cómo se transita entre estados para deducir el estado siguiente (figura A.10).

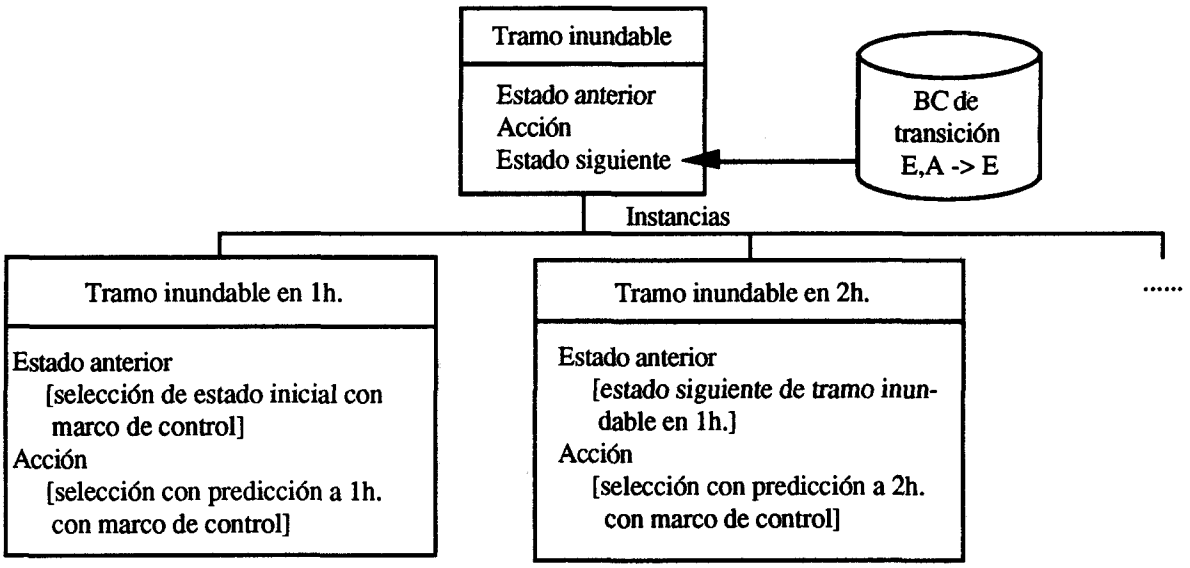


Figura A.10: Clase de Tramo Inundable e instancias para distintos horizontes temporales.

A-VIII. Aspectos generales de interfaz de usuario: operación del SIRAH.

A pesar de que el proceso de simulación empleado por el SIRAH es complejo, como aplicación informática es extremadamente fácil de operar gracias a una interfaz de usuario gráfica basada en productos estandar. De esta forma, el usuario puede obtener resultados del SIRAH sin conocer el modelo a fondo y prácticamente sin usar el teclado. Para operar hace falta únicamente manejar el ratón.

Puesta en marcha del SIRAH.

Para lanzar una ejecución del SIRAH es preciso situarse dentro del entorno proporcionado por el manejador de ventanas de X-windows, crear una ventana de terminal y ubicarse en un directorio que tenga el sirah accesible, escribiendo en la línea de comandos de la shell:

> *sirah <cuenca>*

donde *<cuenca>* es el nombre de un fichero maestro donde tiene su origen la definición de los modelos.

La aplicación SIRAH sigue, en cuanto a interfaz de usuario, las normas de estilo propuestas por la OSF/Motif, siguiendo el paradigma de "señalar_y_pulsar" (point and click). Es decir, todo lo visible puede ser consultado o manipulado, y esto se consigue situando, con el ratón, el cursor sobre el elemento correspondiente y pulsando el botón izquierdo.

Operación del SIRAH.

Una sesión del SIRAH se compone de las siguientes tareas:

- Carga de la base de conocimiento.
- Edición (opción no instrumentada de forma integrada con el entorno).
- Simulación (y consulta).
- Terminación.

La carga de la base de conocimiento se hace una sola vez y debe ser la primera operación una vez dentro del entorno. Una vez efectuada la carga puede editarse y simularse tantas veces como que quiera y en cualquier orden. Finalmente, la terminación es la última tarea que ejecuta el SIRAH.

El panel de control.

Una vez en marcha el SIRAH aparece en la esquina superior derecha de la pantalla un panel con iconos que representan las opciones disponibles en el SIRAH:

- Carga de la base de conocimiento.
- Edición de la red de objetos.
- Simulación.
- Fin.

El panel principal de control consta de cuatro botones etiquetados con iconos representativos de las operaciones correspondientes: una flecha emergiendo de un disco simboliza la carga de la base de conocimiento, un lapicero escribiendo en un papel para representar la edición interactiva, una flecha entre premisas y conclusiones, para simbolizar el proceso de inferencia realizado por la simulación y, por fin, una señal de STOP para terminar (ver figura A.11)[1].

Una sesión normal del SIRAH comprende, en primer lugar, cargar la base de conocimiento, tras lo cual se procede a simular. Una vez que la simulación ha cubierto todas las hipótesis, el usuario puede realizar una

1 La mayoría de las figuras que se muestran son copias directas de terminal gráfico. Por dificultades obvias de edición se presentan al final del anejo.

consulta sobre los resultados de la simulación y -opcionalmente- simular más veces. Por último se puede terminar la sesión.

A continuación se detalla lo que hace cada una de las opciones.

Carga de la base de conocimiento.

La base de conocimiento reside en ficheros de texto, redactada en lenguaje DECON. El fichero maestro de la carga deberá ser el que se ha pasado como argumento al arrancar el SIRAH.

A pesar de que el proceso de carga es bastante eficiente, debido al gran volumen de conocimiento necesario para describir una cuenca con suficiente detalle, la carga es un proceso que lleva un tiempo considerable, del orden de 15 minutos para una base de conocimiento de 6500 reglas (modelo del Jucar) en una máquina HP 9000 de la serie 400.

Para seguir el proceso de carga se muestra al usuario la ventana de la figura A.12. Se muestra el nombre del fichero que se está leyendo, el nombre del objeto que se está leyendo o, si se están leyendo reglas, el último objeto leído, y el número total de reglas cargadas de la base de conocimiento hasta el momento.

La carga de la base de conocimiento es imprescindible para poder ejecutar cualquiera de las opciones siguientes, y es necesario llevarla a cabo solo una vez. Una vez cargados los modelos, si se pulsa el botón de carga no tiene efecto ninguno.

Simulación.

Para activarla se procede pulsando el botón izquierdo del ratón con el cursor situado sobre el icono correspondiente del panel de control. Tras pulsar el icono de simulación aparece una ventana de confirmación, después de lo cual da comienzo empieza el proceso de simulación (figura A.13). Esta se ejecuta en un proceso aparte, distinto del resto, para lo cual el sistema operativo se ve obligado a duplicar el segmento de datos del programa. En ordenadores con poca memoria esto puede llevar algún tiempo (del orden de 20 a 30 segundos) en el que se hace intercambio de páginas de memoria con el disco ("swapping"). Este proceso es atómico por lo que, mientras el sistema operativo realiza esa tarea, no da paso a la ejecución de otros procesos.

El proceso de simulación atraviesa por las siguientes etapas:

- Lectura del estado inicial: Se leen de la red de telemedida todos los datos necesarios para realizar la simulación.

- Generación de escenarios: Se decide qué áreas geográficas es necesario estudiar. Para cada una de ellas se formulan una o varias hipótesis, lo que da lugar a diversos escenarios.
- Simulación de escenarios: se simulan los escenarios basándose en la hipótesis que se ha formulado. Para cada escenario aparece ventanas como las de la figura A.14. Las ventanas de escenario se componen de :
    * *título*: con el nombre del escenario;
    * *área*: contiene un mapa de la cuenca en la parte superior derecha de la ventana, y tiene marcada el área geográfica que corresponde a la hipótesis seleccionada.
    * *detalle*: detalle del área de estudio, donde aparecen los dibujos de los objetos que componen la cuenca según se van simulando o deduciendo. Supone una ampliación de la región marcada en el área.

    Todos los objetos físicos relevantes para la simulación tienen su dibujo asociado; cada dibujo aparece de un color que indica la gravedad de la situación: El código de colores es el siguiente:
    - Verde: situación de normalidad.
    - Amarillo: situación de pre-alerta.
    - Rojo: situación de alerta.

    Los dibujos tienen siempre el mismo tamaño relativo, por lo que si el área de estudio es pequeña, los dibujos aparecerán más grandes. Por ejemplo, el área cubierta por el escenario de la figura A.15 es mucho más grande que el cubierto por la figura A.16.

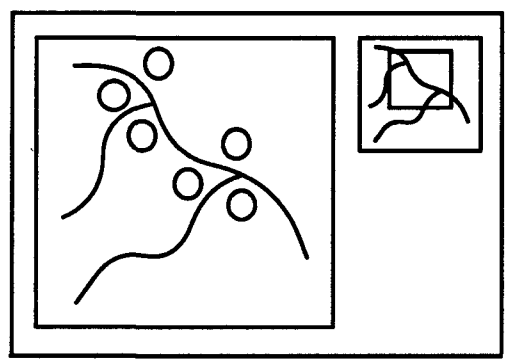


Fig. A.15: área de estudio.

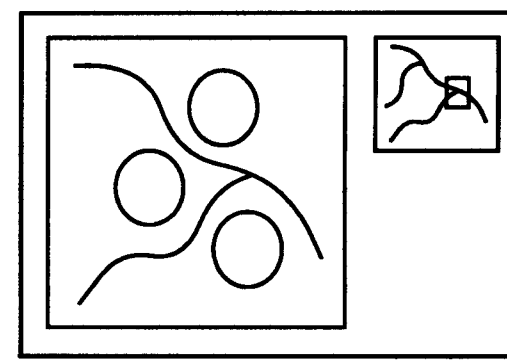


Fig. A.16: área de estudio (zoom).

Los objetos que van apareciendo en un escenario son, y en este orden, Areas Receptoras (circulo), Tramos de Transporte (rectángulo) y Areas Problema (figura A.17).

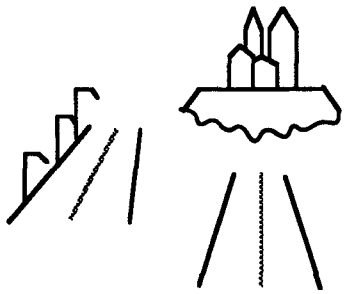


Figura A.17: símbolo de Area Problema.

Durante la simulación de algunos escenarios puede aparecer el mensaje de la figura A.18. La hipótesis es rechazada por el SIRAH en el curso de la simulación, porque no es pertinente o no es verosímil, o ambos criterios conjuntamente.

* *Menú de traza de procesos:* es un menú donde aparecen los nombres de todos los objetos presentes en el escenario, en el mismo orden en que han sido simulados. Posteriormente se puede pedir explicación acerca de ese objeto pulsando sobre el nombre con el botón izquierdo del ratón.
* *Botón "Seleccionar":* pulsando el botón izquierdo del ratón, sobre ese botón aparecen las explicaciones correspondientes al objeto seleccionado en la ventana de detalle o en el menú de traza de proceso.

Presentación de explicaciones.

Una vez simulados todos los escenarios es posible consultar el resultado de la simulación pidiendo información sobre cualquier objeto que aparezca en cualquiera de las ventanas de escenarios. Las explicaciones tienen por objeto mostrar la línea de razonamiento que en cada caso ha seguido el sistema.

Hay dos formas de seleccionar objetos de la ventana de escenarios: a través del menú de traza de proceso o a través de su dibujo. Para seleccionar un objeto a través del menú de traza basta con pulsar dos veces el botón del ratón sobre el nombre del objeto o bien seleccionarlo pulsándolo una vez y pulsando a continuación el botón de "seleccionar". En cualquiera de los dos casos, si el objeto posee representación gráfica se marcará recuadrándose en rojo en el área de detalle.

También es posible seleccionar un objeto pulsando el botón del ratón sobre su dibujo y pulsando el botón de seleccionar. La respuesta a la selección de un objeto es la aparición de una ventana de explicación.

Hay dos tipos de objetos en el SIRAH: marcos y tareas. Los marcos son los objetos más simples que maneja el SIRAH y representan unidades cognitivas sobre las que se realiza razonamiento superficial mediante reglas y métodos ad hoc. Son marcos las áreas problema: carreteras, poblaciones, entre otros no visibles por el interfaz de

usuario. Cuando se selecciona un marco, la ventana que aparece es como la mostrada en la figura A.19. Se compone de una barra de título donde dice el nombre del marco y el escenario en que está, junto con el botón de salir. Al pulsar este botón desaparece la ventana con el marco.

En la región de texto de una traza de marco viene la siguiente información:

* FUENTE: es el nombre del objeto.
* VARIABLES: son los atributos del objeto con sus valores.

La representación de objetos físicos se realiza caracterizándolos por atributos o variables que pueden tomar distintos valores (posiblemente más de uno) calculada mediante un método especial o bien por imposición directa del modelizador. Un método a destacar de entre todos los disponibles es "Reglas" que hace que se calcule la variable por inferencia mediante una base de reglas. Para los atributos calculados por reglas se ofrece una explicación limitada del porqué de su valor listando a continuación todas las reglas que se han disparado contribuyendo a su valor. Por razones de concisión sólo se muestran las reglas que han concluido directamente sobre el atributo, excluyéndose las que concluyen sobre los atributos que participan en las premisas y las que concluyen sobre atributos que participan en las premisas de estos, etc. En la figura se puede ver la consulta a un área problema.

Para las Tareas Básicas de Simulación no se han instrumentado mecanismos de explicación todavía. Se presenta únicamente una visión tipo "caja negra", mostrando un marco con entradas y salidas como el de la figura la figura A.20. La explicación tiene tres partes: la FUENTE que denota el objeto representado, las ENTRADAS, donde se muestran los atributos de entrada y sus valores y las SALIDAS, con los atributos de salida y sus valores.

Frecuentemente, los valores de las entradas y las salidas no admitirán una descripción en texto por estar representadas mediante series temporales. En este caso los valores se representan en la parte del texto con el mismo nombre del atributo, seguido por un número de orden entre paréntesis, de manera que se pueden identificar las ventanas gráficas que se corresponden con el valor del atributo y que aparecen junto a la ventana del texto.

Las ventanas de serie temporal están supeditadas a las ventanas de texto del objeto a que pertenecen de manera que cuando se destruyen estos, desaparecen también las series temporales. Asimismo, las ventanas de explicación están supeditadas al escenario de manera que cuando se iconifica o destruye éste, se iconifica o destruyen las ventanas de explicación.

Figura A.11: El panel de control.

Figura A.12: Plantilla de carga.

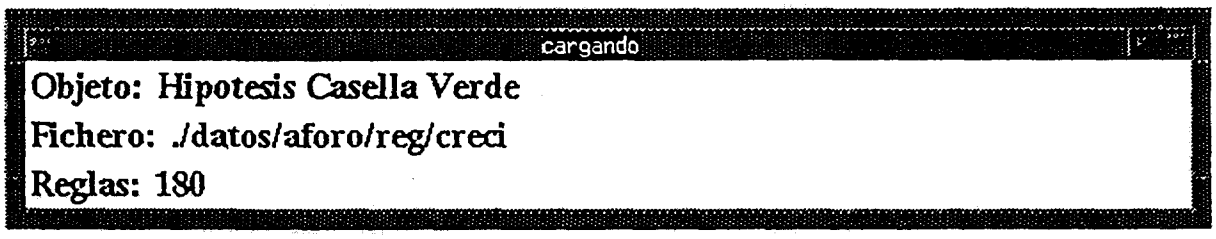
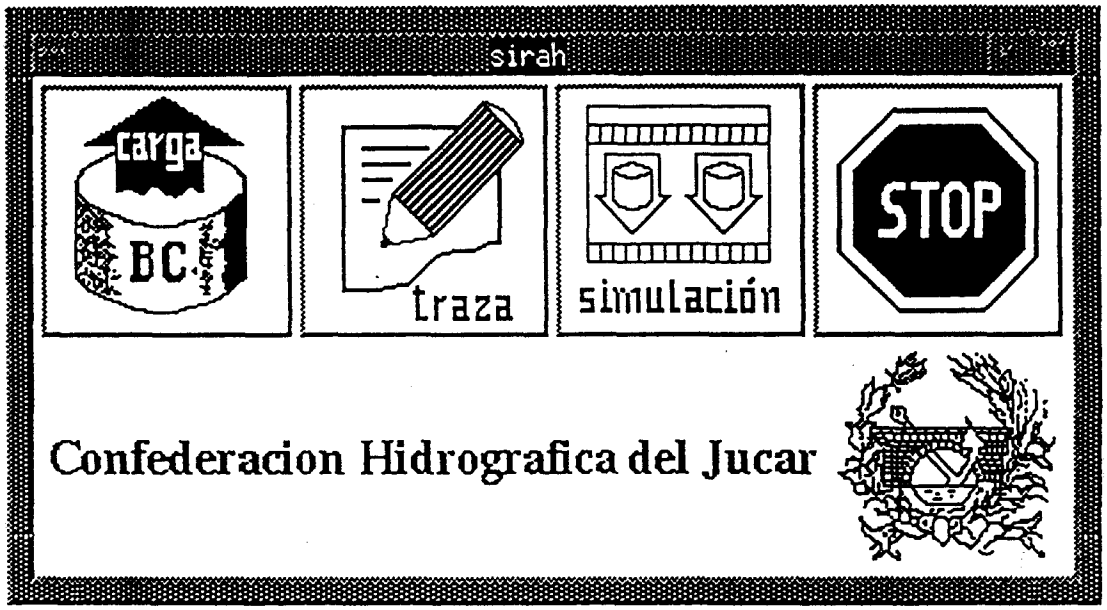


Figura A.13: Activación de una simulación.



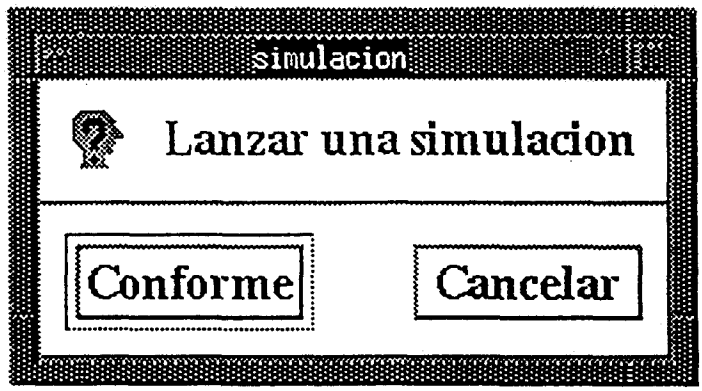


Figura A.14: Escenarios de simulación.

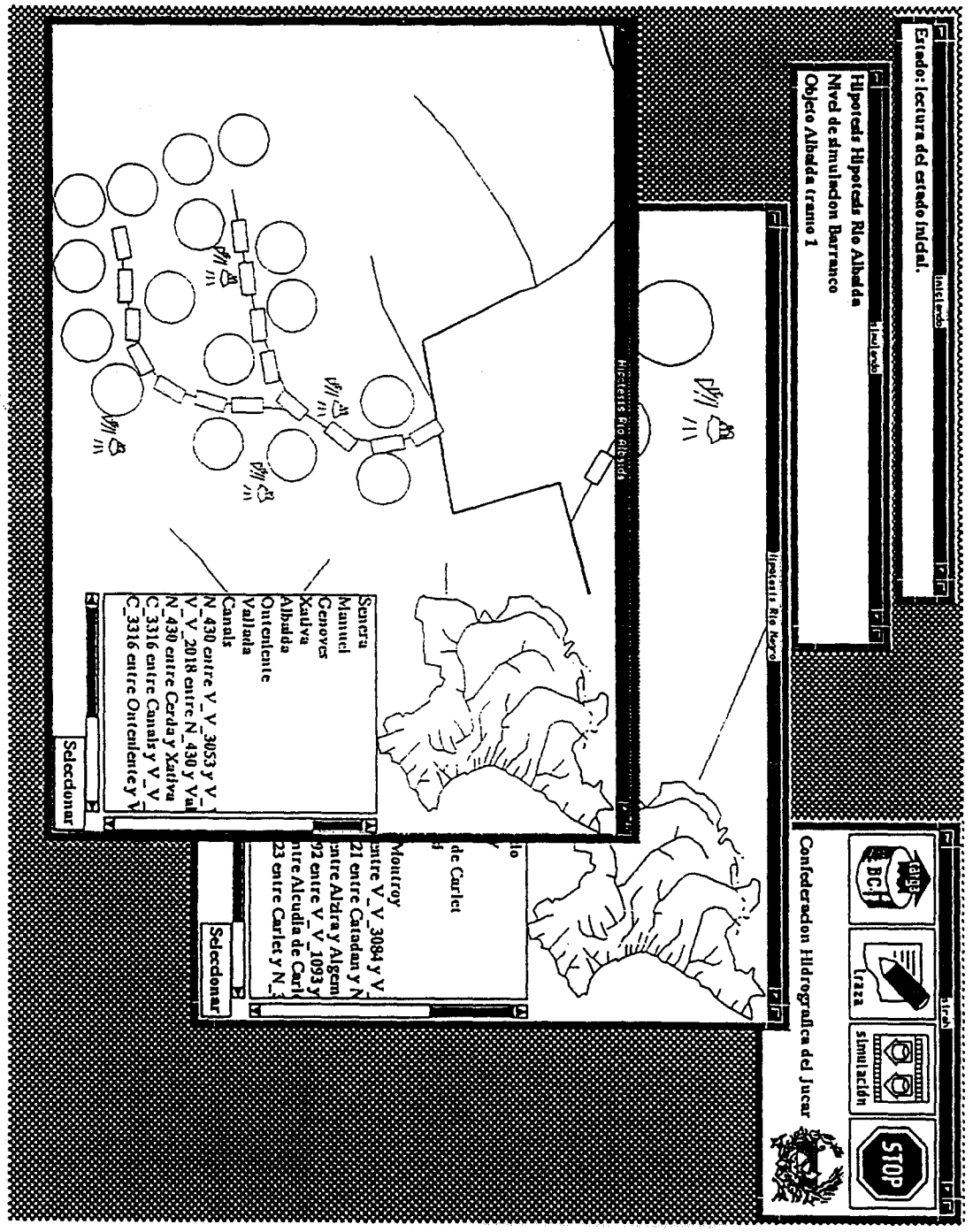


Figura A.18: Mensaje de rechazo de escenario.

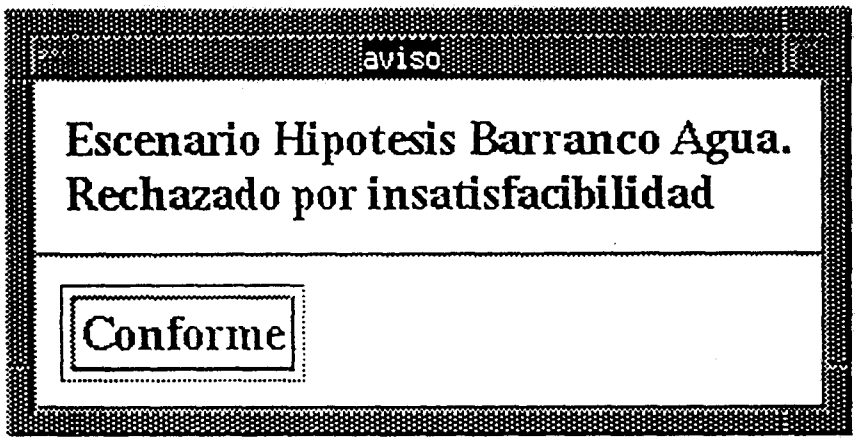


Figura A.19: Ventana de traza de un marco.

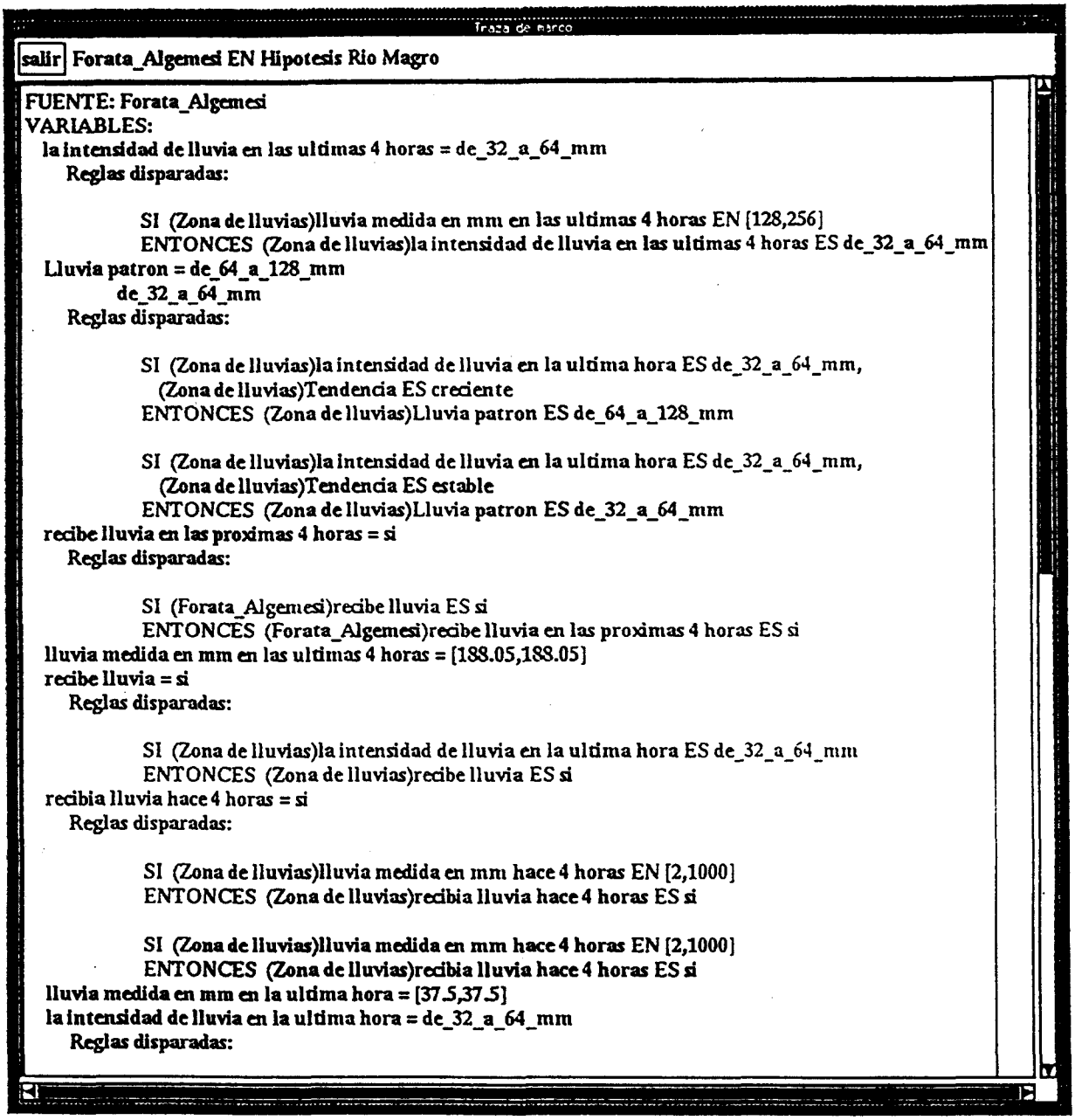


Figura A.20: Ventanas de traza de una TBS.